Incremental FX Test Cases

Timothy Daly

January 5, 2007

Contents

Abstract

We can test a large portion of the current FX system using the linear bubble sort program. This whitepaper examines various permutations of the sort to show a systematic set of changes and their potential FX results.

1 Overview

We need to be able to test FX in a systematic manner. This can be done quite easily by starting with the linear bubblesort problem and making tiny changes to the program to test a single new feature. We have several general classes of change we need to explore including change of:

- Data Stride
- Representation
- Control Flow
- System Libraries
- System Input
- Repository Features
- Platform
- Control Structures
- Algorithm

In this whitepaper I present a sequence of trivial programs that are intended to exercise single features of FX so we can isolate and explore our ability to handle each feature.

Along the way I also introduce problems for Face and the repository.

2 A: Initial Problem

This initial problem to consider the problem of doing a bubble sort. The first case, which we cannot yet handle, is the bubble sort expressed as a loop. The second case, which we should be able to handle is an "unrolled version" of the same code.

2.1 Case A1: The Bubble Sort

This is a standard bubble sort of 10 data items. The bubble sort does not input or output and it works on a fixed length array of doublewords.

⟨*CaseA1*⟩≡

```
#include <stdio.h>
int main()
{ int a[10] = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9};
  int swap = 1; /* true */
  int i = 0;
  int n = 9;
  int temp = 0;
  while (swap == 1) {
    swap = 0;
    i = 0;
    while (i < n) {
      if (a[i] < a[i+1]) {
        temp = a[i+1];
        a[i+1] = a[i];
        a[i] = temp;
        swap = 1;
      }
      i = i+1;
    }
  }
}
```

2.2 Case A2: A Linear Bubble Sort

Here we have rewritten the loop into a linear form so we do not have to compute the action of the loop.

In particular, we are careful to only use conditional statements and a set of statements equivalent to a swap. The swap only uses a temporary variable. We do no input or output. There are a fixed number of items to sort. The data is uniform in shape, size, and stride.

2.2.1 Face Issue

We should be able to select a particular subset of lines and rerun the analysis on just that subset.

⟨*CaseA2*⟩≡

```
#include <stdio.h>
int main()
{ int a[10] = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9};
  int tmp;
  /* carry the highest number to the end each time */
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
  /* { 2, 4, 6, 8, 1, 10, 3, 5, 7, 9}; */
  if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
  /* { 2, 4, 6, 8, 1, 3, 10, 5, 7, 9}; */
  if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
  /* { 2, 4, 6, 8, 1, 3, 5, 10, 7, 9}; */
  if (a[7] > a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 10, 9}; */
  if (a[8] > a[9]) { tmp = a[9] ; a[9] = a[8]; a[8] = tmp; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */

  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
```

```
/* { 2, 4, 6, 1, 8, 3, 5, 7, 9, 10}; */
if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 2, 4, 6, 1, 3, 8, 5, 7, 9, 10}; */
if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
/* { 2, 4, 6, 1, 3, 5, 8, 7, 9, 10}; */
if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (a[7] > a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
/* we don't have to test the last case because a[9] is sorted */

if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 2, 4, 1, 6, 3, 5, 7, 8, 9, 10}; */
if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 2, 4, 1, 3, 6, 5, 7, 8, 9, 10}; */
if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
/* a[8] and a[9] must be sorted */

if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 2, 1, 4, 3, 5, 6, 7, 8, 9, 10}; */
if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */

if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
```

```
  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

}
```

3 B: Change of Data Stride

Stride is the number of bytes taken by each data element. We started out using integers which take up 4 bytes per number. Here we start to look at other sizes to see how this changes the CCAs.

3.1 Case B1: Integer to Char

This case uses a data stride of a single byte, which is the internal representation of the character.

⟨*CaseB1*⟩≡

```
#include <stdio.h>
int main()
{ char a[10] = { 'b', 'd', 'f', 'h', 'j', 'a', 'c', 'e', 'g', 'i'};
  char tmp;
  /* carry the highest number to the end each time */
  /* { 'b', 'd', 'f', 'h', 'j', 'a', 'c', 'e', 'g', 'i'}; */
  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 'b', 'd', 'f', 'h', 'j', 'a', 'c', 'e', 'g', 'i'}; */
  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 'b', 'd', 'f', 'h', 'j', 'a', 'c', 'e', 'g', 'i'}; */
  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 'b', 'd', 'f', 'h', 'j', 'a', 'c', 'e', 'g', 'i'}; */
  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 'b', 'd', 'f', 'h', 'j', 'a', 'c', 'e', 'g', 'i'}; */
  if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
  /* { 'b', 'd', 'f', 'h', 'a', 'j', 'c', 'e', 'g', 'i'}; */
  if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
  /* { 'b', 'd', 'f', 'h', 'a', 'c', 'j', 'e', 'g', 'i'}; */
  if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
  /* { 'b', 'd', 'f', 'h', 'a', 'c', 'e', 'j', 'g', 'i'}; */
  if (a[7] > a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
  /* { 'b', 'd', 'f', 'h', 'a', 'c', 'e', 'g', 'j', 'i'}; */
  if (a[8] > a[9]) { tmp = a[9] ; a[9] = a[8]; a[8] = tmp; }
  /* { 'b', 'd', 'f', 'h', 'a', 'c', 'e', 'g', 'i', 'j'}; */

  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 'b', 'd', 'f', 'h', 'a', 'c', 'e', 'g', 'i', 'j'}; */
  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 'b', 'd', 'f', 'h', 'a', 'c', 'e', 'g', 'i', 'j'}; */
  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 'b', 'd', 'f', 'h', 'a', 'c', 'e', 'g', 'i', 'j'}; */
  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 'b', 'd', 'f', 'a', 'h', 'c', 'e', 'g', 'i', 'j'}; */
  if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
  /* { 'b', 'd', 'f', 'a', 'c', 'h', 'e', 'g', 'i', 'j'}; */
```

```
if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
/* { 'b', 'd', 'f', 'a', 'c', 'e', 'h', 'g', 'i', 'j'}; */
if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
/* { 'b', 'd', 'f', 'a', 'c', 'e', 'g', 'h', 'i', 'j'}; */
if (a[7] > a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
/* { 'b', 'd', 'f', 'a', 'c', 'e', 'g', 'h', 'i', 'j'}; */
/* we don't have to test the last case because a[9] is sorted */

if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 'b', 'd', 'f', 'a', 'c', 'e', 'g', 'h', 'i', 'j'}; */
if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 'b', 'd', 'f', 'a', 'c', 'e', 'g', 'h', 'i', 'j'}; */
if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 'b', 'd', 'a', 'f', 'c', 'e', 'g', 'h', 'i', 'j'}; */
if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 'b', 'd', 'a', 'c', 'f', 'e', 'g', 'h', 'i', 'j'}; */
if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 'b', 'd', 'a', 'c', 'e', 'f', 'g', 'h', 'i', 'j'}; */
if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
/* { 'b', 'd', 'a', 'c', 'e', 'f', 'g', 'h', 'i', 'j'}; */
if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
/* { 'b', 'd', 'a', 'c', 'e', 'f', 'g', 'h', 'i', 'j'}; */
/* a[8] and a[9] must be sorted */

if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 'b', 'd', 'a', 'c', 'e', 'f', 'g', 'h', 'i', 'j'}; */
if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 'b', 'a', 'd', 'c', 'e', 'f', 'g', 'h', 'i', 'j'}; */
if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 'b', 'a', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */
if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 'b', 'a', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */
if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 'b', 'a', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */
if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
/* { 'b', 'a', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */

if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */
if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */
if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */
if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */
if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
```

```
  /* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */

  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */
  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */
  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */
  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */

  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */
  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */
  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */

  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */
  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */

  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}; */

}
```

3.2 Case B2: Integer to Float

This changes the stride to use small floating point point numbers which are represented as doubleword but use different machine instructions for manipulation.

⟨*CaseB2*⟩≡

```
#include <stdio.h>
int main()
{ float a[10] = { 2., 4., 6., 8., 10., 1., 3., 5., 7., 9.};
  float tmp;
  /* carry the highest number to the end each time */
  /* { 2., 4., 6., 8., 10., 1., 3., 5., 7., 9.}; */
  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 2., 4., 6., 8., 10., 1., 3., 5., 7., 9.}; */
  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 2., 4., 6., 8., 10., 1., 3., 5., 7., 9.}; */
  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 2., 4., 6., 8., 10., 1., 3., 5., 7., 9.}; */
  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 2., 4., 6., 8., 10., 1., 3., 5., 7., 9.}; */
  if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
  /* { 2., 4., 6., 8., 1., 10., 3., 5., 7., 9.}; */
  if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
  /* { 2., 4., 6., 8., 1., 3., 10., 5., 7., 9.}; */
  if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
  /* { 2., 4., 6., 8., 1., 3., 5., 10., 7., 9.}; */
  if (a[7] > a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
  /* { 2., 4., 6., 8., 1., 3., 5., 7., 10., 9.}; */
  if (a[8] > a[9]) { tmp = a[9] ; a[9] = a[8]; a[8] = tmp; }
  /* { 2., 4., 6., 8., 1., 3., 5., 7., 9., 10.}; */

  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 2., 4., 6., 8., 1., 3., 5., 7., 9., 10.}; */
  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 2., 4., 6., 8., 1., 3., 5., 7., 9., 10.}; */
  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 2., 4., 6., 8., 1., 3., 5., 7., 9., 10.}; */
  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 2., 4., 6., 1., 8., 3., 5., 7., 9., 10.}; */
  if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
  /* { 2., 4., 6., 1., 3., 8., 5., 7., 9., 10.}; */
  if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
  /* { 2., 4., 6., 1., 3., 5., 8., 7., 9., 10.}; */
  if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
  /* { 2., 4., 6., 1., 3., 5., 7., 8., 9., 10.}; */
  if (a[7] > a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
  /* { 2., 4., 6., 1., 3., 5., 7., 8., 9., 10.}; */
```

```
/* we don't have to test the last case because a[9] is sorted */

if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 2., 4., 6., 1., 3., 5., 7., 8., 9., 10.}; */
if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 2., 4., 6., 1., 3., 5., 7., 8., 9., 10.}; */
if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 2., 4., 1., 6., 3., 5., 7., 8., 9., 10.}; */
if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 2., 4., 1., 3., 6., 5., 7., 8., 9., 10.}; */
if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 2., 4., 1., 3., 5., 6., 7., 8., 9., 10.}; */
if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
/* { 2., 4., 1., 3., 5., 6., 7., 8., 9., 10.}; */
if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
/* { 2., 4., 1., 3., 5., 6., 7., 8., 9., 10.}; */
/* a[8] and a[9] must be sorted */

if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 2., 4., 1., 3., 5., 6., 7., 8., 9., 10.}; */
if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 2., 1., 4., 3., 5., 6., 7., 8., 9., 10.}; */
if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 2., 1., 3., 4., 5., 6., 7., 8., 9., 10.}; */
if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 2., 1., 3., 4., 5., 6., 7., 8., 9., 10.}; */
if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 2., 1., 3., 4., 5., 6., 7., 8., 9., 10.}; */
if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
/* { 2., 1., 3., 4., 5., 6., 7., 8., 9., 10.}; */

if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */
if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */
if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */
if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */
if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */

if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */
if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */
```

```
  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */
  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */

  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */
  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */
  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */

  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */
  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */

  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.}; */

}
```

4 C: Change of Representation

This second changes the way we stride across memory in a different form. Here we are stepping along pointers to data rather than stepping along the data itself.

4.1 Case C1: Array to List

⟨*CaseC1*⟩≡

5 D: Change of Control Flow

The initial linear bubble sort routine uses the > compare function. Here we "compare and contrast" the CCAs that result if we change the compare function but nothing else.

5.1 Case D1: Partial Symmetric Comparitors > vs <

This case looks at changing from using > to <. It ignores the equality case.

⟨*CaseD1*⟩≡

```
#include <stdio.h>
int main()
{ int a[10] = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9};
  int tmp;
  /* carry the highest number to the end each time */
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[0] < a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 4, 2, 6, 8, 10, 1, 3, 5, 7, 9, } */
  if (a[1] < a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 4, 6, 2, 8, 10, 1, 3, 5, 7, 9, } */
  if (a[2] < a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 4, 6, 8, 2, 10, 1, 3, 5, 7, 9, } */
  if (a[3] < a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 4, 6, 8, 10, 2, 1, 3, 5, 7, 9, } */
  if (a[4] < a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
  /* { 4, 6, 8, 10, 2, 1, 3, 5, 7, 9, } */
  if (a[5] < a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
  /* { 4, 6, 8, 10, 2, 3, 1, 5, 7, 9, } */
  if (a[6] < a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
  /* { 4, 6, 8, 10, 2, 3, 5, 1, 7, 9, } */
  if (a[7] < a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
  /* { 4, 6, 8, 10, 2, 3, 5, 7, 1, 9, } */
  if (a[8] < a[9]) { tmp = a[9] ; a[9] = a[8]; a[8] = tmp; }
  /* { 4, 6, 8, 10, 2, 3, 5, 7, 9, 1, } */

  if (a[0] < a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 6, 4, 8, 10, 2, 3, 5, 7, 9, 1, } */
  if (a[1] < a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 6, 8, 4, 10, 2, 3, 5, 7, 9, 1, } */
  if (a[2] < a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 6, 8, 10, 4, 2, 3, 5, 7, 9, 1, } */
  if (a[3] < a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 6, 8, 10, 4, 2, 3, 5, 7, 9, 1, } */
  if (a[4] < a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
  /* { 6, 8, 10, 4, 3, 2, 5, 7, 9, 1, } */
  if (a[5] < a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
```

```
/* { 6, 8, 10, 4, 3, 5, 2, 7, 9, 1, } */
if (a[6] < a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
/* { 6, 8, 10, 4, 3, 5, 7, 2, 9, 1, } */
if (a[7] < a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
/* { 6, 8, 10, 4, 3, 5, 7, 9, 2, 1, } */
/* we don't have to test the last case because a[9] is sorted */

if (a[0] < a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 8, 6, 10, 4, 3, 5, 7, 9, 2, 1, } */
if (a[1] < a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 8, 10, 6, 4, 3, 5, 7, 9, 2, 1, } */
if (a[2] < a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 8, 10, 6, 4, 3, 5, 7, 9, 2, 1, } */
if (a[3] < a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 8, 10, 6, 4, 3, 5, 7, 9, 2, 1, } */
if (a[4] < a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 8, 10, 6, 4, 5, 3, 7, 9, 2, 1, } */
if (a[5] < a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
/* { 8, 10, 6, 4, 5, 7, 3, 9, 2, 1, } */
if (a[6] < a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
/* { 8, 10, 6, 4, 5, 7, 9, 3, 2, 1, } */
/* a[8] and a[9] must be sorted */

if (a[0] < a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 10, 8, 6, 4, 5, 7, 9, 3, 2, 1, } */
if (a[1] < a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 10, 8, 6, 4, 5, 7, 9, 3, 2, 1, } */
if (a[2] < a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 10, 8, 6, 4, 5, 7, 9, 3, 2, 1, } */
if (a[3] < a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 10, 8, 6, 5, 4, 7, 9, 3, 2, 1, } */
if (a[4] < a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 10, 8, 6, 5, 7, 4, 9, 3, 2, 1, } */
if (a[5] < a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
/* { 10, 8, 6, 5, 7, 9, 4, 3, 2, 1, } */

if (a[0] < a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 10, 8, 6, 5, 7, 9, 4, 3, 2, 1, } */
if (a[1] < a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 10, 8, 6, 5, 7, 9, 4, 3, 2, 1, } */
if (a[2] < a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 10, 8, 6, 5, 7, 9, 4, 3, 2, 1, } */
if (a[3] < a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 10, 8, 6, 7, 5, 9, 4, 3, 2, 1, } */
if (a[4] < a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 10, 8, 6, 7, 9, 5, 4, 3, 2, 1, } */
```

```
  if (a[0] < a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 10, 8, 6, 7, 9, 5, 4, 3, 2, 1, } */
  if (a[1] < a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 10, 8, 6, 7, 9, 5, 4, 3, 2, 1, } */
  if (a[2] < a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 10, 8, 7, 6, 9, 5, 4, 3, 2, 1, } */
  if (a[3] < a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 10, 8, 7, 9, 6, 5, 4, 3, 2, 1, } */

  if (a[0] < a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 10, 8, 7, 9, 6, 5, 4, 3, 2, 1, } */
  if (a[1] < a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 10, 8, 7, 9, 6, 5, 4, 3, 2, 1, } */
  if (a[2] < a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 10, 8, 9, 7, 6, 5, 4, 3, 2, 1, } */

  if (a[0] < a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 10, 8, 9, 7, 6, 5, 4, 3, 2, 1, } */
  if (a[1] < a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, } */

  if (a[0] < a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, } */

}
```

5.2 Case D2: Asymmetric Comparitors > vs <=

This case handles the equality case in the compare. We need to contrast this with the initial linear bubble sort to see how the CCAs change.

⟨*CaseD2*⟩≡

```
#include <stdio.h>
int main()
{ int a[10] = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9};
  int tmp;
  int i;
  /* carry the highest number to the end each time */
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[0] <= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 4, 2, 6, 8, 10, 1, 3, 5, 7, 9, } */
  if (a[1] <= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 4, 6, 2, 8, 10, 1, 3, 5, 7, 9, } */
  if (a[2] <= a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 4, 6, 8, 2, 10, 1, 3, 5, 7, 9, } */
  if (a[3] <= a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 4, 6, 8, 10, 2, 1, 3, 5, 7, 9, } */
  if (a[4] <= a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
  /* { 4, 6, 8, 10, 2, 1, 3, 5, 7, 9, } */
  if (a[5] <= a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
  /* { 4, 6, 8, 10, 2, 3, 1, 5, 7, 9, } */
  if (a[6] <= a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
  /* { 4, 6, 8, 10, 2, 3, 5, 1, 7, 9, } */
  if (a[7] <= a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
  /* { 4, 6, 8, 10, 2, 3, 5, 7, 1, 9, } */
  if (a[8] <= a[9]) { tmp = a[9] ; a[9] = a[8]; a[8] = tmp; }
  /* { 4, 6, 8, 10, 2, 3, 5, 7, 9, 1, } */

  if (a[0] <= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 6, 4, 8, 10, 2, 3, 5, 7, 9, 1, } */
  if (a[1] <= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 6, 8, 4, 10, 2, 3, 5, 7, 9, 1, } */
  if (a[2] <= a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 6, 8, 10, 4, 2, 3, 5, 7, 9, 1, } */
  if (a[3] <= a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 6, 8, 10, 4, 2, 3, 5, 7, 9, 1, } */
  if (a[4] <= a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
  /* { 6, 8, 10, 4, 3, 2, 5, 7, 9, 1, } */
  if (a[5] <= a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
  /* { 6, 8, 10, 4, 3, 5, 2, 7, 9, 1, } */
  if (a[6] <= a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
  /* { 6, 8, 10, 4, 3, 5, 7, 2, 9, 1, } */
  if (a[7] <= a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
```

```
/* { 6, 8, 10, 4, 3, 5, 7, 9, 2, 1, } */
/* we don't have to test the last case because a[9] is sorted */

if (a[0] <= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 8, 6, 10, 4, 3, 5, 7, 9, 2, 1, } */
if (a[1] <= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 8, 10, 6, 4, 3, 5, 7, 9, 2, 1, } */
if (a[2] <= a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 8, 10, 6, 4, 3, 5, 7, 9, 2, 1, } */
if (a[3] <= a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 8, 10, 6, 4, 3, 5, 7, 9, 2, 1, } */
if (a[4] <= a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 8, 10, 6, 4, 5, 3, 7, 9, 2, 1, } */
if (a[5] <= a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
/* { 8, 10, 6, 4, 5, 7, 3, 9, 2, 1, } */
if (a[6] <= a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
/* { 8, 10, 6, 4, 5, 7, 9, 3, 2, 1, } */
/* a[8] and a[9] must be sorted */

if (a[0] <= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 10, 8, 6, 4, 5, 7, 9, 3, 2, 1, } */
if (a[1] <= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 10, 8, 6, 4, 5, 7, 9, 3, 2, 1, } */
if (a[2] <= a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 10, 8, 6, 4, 5, 7, 9, 3, 2, 1, } */
if (a[3] <= a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 10, 8, 6, 5, 4, 7, 9, 3, 2, 1, } */
if (a[4] <= a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 10, 8, 6, 5, 7, 4, 9, 3, 2, 1, } */
if (a[5] <= a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
/* { 10, 8, 6, 5, 7, 9, 4, 3, 2, 1, } */

if (a[0] <= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 10, 8, 6, 5, 7, 9, 4, 3, 2, 1, } */
if (a[1] <= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 10, 8, 6, 5, 7, 9, 4, 3, 2, 1, } */
if (a[2] <= a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 10, 8, 6, 5, 7, 9, 4, 3, 2, 1, } */
if (a[3] <= a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 10, 8, 6, 7, 5, 9, 4, 3, 2, 1, } */
if (a[4] <= a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 10, 8, 6, 7, 9, 5, 4, 3, 2, 1, } */

if (a[0] <= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 10, 8, 6, 7, 9, 5, 4, 3, 2, 1, } */
if (a[1] <= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
```

```
  /* { 10, 8, 6, 7, 9, 5, 4, 3, 2, 1, } */
  if (a[2] <= a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 10, 8, 7, 6, 9, 5, 4, 3, 2, 1, } */
  if (a[3] <= a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 10, 8, 7, 9, 6, 5, 4, 3, 2, 1, } */

  if (a[0] <= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 10, 8, 7, 9, 6, 5, 4, 3, 2, 1, } */
  if (a[1] <= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 10, 8, 7, 9, 6, 5, 4, 3, 2, 1, } */
  if (a[2] <= a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 10, 8, 9, 7, 6, 5, 4, 3, 2, 1, } */

  if (a[0] <= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 10, 8, 9, 7, 6, 5, 4, 3, 2, 1, } */
  if (a[1] <= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, } */

  if (a[0] <= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, } */

}
```

5.3 Case D3: Symmetric Comparitors >= vs <=

This case handles the equality case in the compare. We need to contrast this with the initial linear bubble sort to see how the CCAs change.

⟨*CaseD3*⟩≡

```
#include <stdio.h>
int main()
{ int a[10] = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9};
  int tmp;
  int i;
  /* carry the highest number to the end each time */
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[0] >= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9, } */
  if (a[1] >= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9, } */
  if (a[2] >= a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9, } */
  if (a[3] >= a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9, } */
  if (a[4] >= a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
  /* { 2, 4, 6, 8, 1, 10, 3, 5, 7, 9, } */
  if (a[5] >= a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
  /* { 2, 4, 6, 8, 1, 3, 10, 5, 7, 9, } */
  if (a[6] >= a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
  /* { 2, 4, 6, 8, 1, 3, 5, 10, 7, 9, } */
  if (a[7] >= a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 10, 9, } */
  if (a[8] >= a[9]) { tmp = a[9] ; a[9] = a[8]; a[8] = tmp; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10, } */

  if (a[0] >= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10, } */
  if (a[1] >= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10, } */
  if (a[2] >= a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10, } */
  if (a[3] >= a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 2, 4, 6, 1, 8, 3, 5, 7, 9, 10, } */
  if (a[4] >= a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
  /* { 2, 4, 6, 1, 3, 8, 5, 7, 9, 10, } */
  if (a[5] >= a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
  /* { 2, 4, 6, 1, 3, 5, 8, 7, 9, 10, } */
  if (a[6] >= a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
  /* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10, } */
  if (a[7] >= a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
```

```
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10, } */
/* we don't have to test the last case because a[9] is sorted */

if (a[0] >= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10, } */
if (a[1] >= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10, } */
if (a[2] >= a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 2, 4, 1, 6, 3, 5, 7, 8, 9, 10, } */
if (a[3] >= a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 2, 4, 1, 3, 6, 5, 7, 8, 9, 10, } */
if (a[4] >= a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10, } */
if (a[5] >= a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10, } */
if (a[6] >= a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10, } */
/* a[8] and a[9] must be sorted */

if (a[0] >= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10, } */
if (a[1] >= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 2, 1, 4, 3, 5, 6, 7, 8, 9, 10, } */
if (a[2] >= a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10, } */
if (a[3] >= a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10, } */
if (a[4] >= a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10, } */
if (a[5] >= a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10, } */

if (a[0] >= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */
if (a[1] >= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */
if (a[2] >= a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */
if (a[3] >= a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */
if (a[4] >= a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */

if (a[0] >= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */
if (a[1] >= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
```

```
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */
  if (a[2] >= a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */
  if (a[3] >= a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */

  if (a[0] >= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */
  if (a[1] >= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */
  if (a[2] >= a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */

  if (a[0] >= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */
  if (a[1] >= a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */

  if (a[0] >= a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, } */

}
```

5.4 Case D4: Swap with Boolean Flag

We return to the original linear sort routine.

Here we introduce a simple integer flag. The flag is set to 1 ("true") if we need to swap elements and then reset to 0 ("false"). This is in preparation for the next steps of control flow.

Because we only manipulate the swp flag if a swap occurs we should be able to find a case in the flow control where the swp flag is never changed. This case would occur if the array is already sorted.

If we find this case we have tested that we are not sensitive to the data in the array. If we don't find this case then we might be passing along data sensitive information.

⟨*CaseD4*⟩≡

```
#include <stdio.h>
int main()
{ int a[10] = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9};
  int tmp;
  int swp;
  swp = 0;
  /* carry the highest number to the end each time */
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[0] > a[1]) { swp=1; tmp = a[1] ; a[1] = a[0]; a[0] = tmp; swp=0; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[1] > a[2]) { swp=1; tmp = a[2] ; a[2] = a[1]; a[1] = tmp; swp=0; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[2] > a[3]) { swp=1; tmp = a[3] ; a[3] = a[2]; a[2] = tmp; swp=0; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[3] > a[4]) { swp=1; tmp = a[4] ; a[4] = a[3]; a[3] = tmp; swp=0; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[4] > a[5]) { swp=1; tmp = a[5] ; a[5] = a[4]; a[4] = tmp; swp=0; }
  /* { 2, 4, 6, 8, 1, 10, 3, 5, 7, 9}; */
  if (a[5] > a[6]) { swp=1; tmp = a[6] ; a[6] = a[5]; a[5] = tmp; swp=0; }
  /* { 2, 4, 6, 8, 1, 3, 10, 5, 7, 9}; */
  if (a[6] > a[7]) { swp=1; tmp = a[7] ; a[7] = a[6]; a[6] = tmp; swp=0; }
  /* { 2, 4, 6, 8, 1, 3, 5, 10, 7, 9}; */
  if (a[7] > a[8]) { swp=1; tmp = a[8] ; a[8] = a[7]; a[7] = tmp; swp=0; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 10, 9}; */
  if (a[8] > a[9]) { swp=1; tmp = a[9] ; a[9] = a[8]; a[8] = tmp; swp=0; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */

  if (a[0] > a[1]) { swp=1; tmp = a[1] ; a[1] = a[0]; a[0] = tmp; swp=0; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
  if (a[1] > a[2]) { swp=1; tmp = a[2] ; a[2] = a[1]; a[1] = tmp; swp=0; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
  if (a[2] > a[3]) { swp=1; tmp = a[3] ; a[3] = a[2]; a[2] = tmp; swp=0; }
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
```

```
if (a[3] > a[4]) { swp=1; tmp = a[4] ; a[4] = a[3]; a[3] = tmp; swp=0; }
/* { 2, 4, 6, 1, 8, 3, 5, 7, 9, 10}; */
if (a[4] > a[5]) { swp=1; tmp = a[5] ; a[5] = a[4]; a[4] = tmp; swp=0; }
/* { 2, 4, 6, 1, 3, 8, 5, 7, 9, 10}; */
if (a[5] > a[6]) { swp=1; tmp = a[6] ; a[6] = a[5]; a[5] = tmp; swp=0; }
/* { 2, 4, 6, 1, 3, 5, 8, 7, 9, 10}; */
if (a[6] > a[7]) { swp=1; tmp = a[7] ; a[7] = a[6]; a[6] = tmp; swp=0; }
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (a[7] > a[8]) { swp=1; tmp = a[8] ; a[8] = a[7]; a[7] = tmp; swp=0; }
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
/* we don't have to test the last case because a[9] is sorted */

if (a[0] > a[1]) { swp=1; tmp = a[1] ; a[1] = a[0]; a[0] = tmp; swp=0; }
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (a[1] > a[2]) { swp=1; tmp = a[2] ; a[2] = a[1]; a[1] = tmp; swp=0; }
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (a[2] > a[3]) { swp=1; tmp = a[3] ; a[3] = a[2]; a[2] = tmp; swp=0; }
/* { 2, 4, 1, 6, 3, 5, 7, 8, 9, 10}; */
if (a[3] > a[4]) { swp=1; tmp = a[4] ; a[4] = a[3]; a[3] = tmp; swp=0; }
/* { 2, 4, 1, 3, 6, 5, 7, 8, 9, 10}; */
if (a[4] > a[5]) { swp=1; tmp = a[5] ; a[5] = a[4]; a[4] = tmp; swp=0; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (a[5] > a[6]) { swp=1; tmp = a[6] ; a[6] = a[5]; a[5] = tmp; swp=0; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (a[6] > a[7]) { swp=1; tmp = a[7] ; a[7] = a[6]; a[6] = tmp; swp=0; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
/* a[8] and a[9] must be sorted */

if (a[0] > a[1]) { swp=1; tmp = a[1] ; a[1] = a[0]; a[0] = tmp; swp=0; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (a[1] > a[2]) { swp=1; tmp = a[2] ; a[2] = a[1]; a[1] = tmp; swp=0; }
/* { 2, 1, 4, 3, 5, 6, 7, 8, 9, 10}; */
if (a[2] > a[3]) { swp=1; tmp = a[3] ; a[3] = a[2]; a[2] = tmp; swp=0; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[3] > a[4]) { swp=1; tmp = a[4] ; a[4] = a[3]; a[3] = tmp; swp=0; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[4] > a[5]) { swp=1; tmp = a[5] ; a[5] = a[4]; a[4] = tmp; swp=0; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[5] > a[6]) { swp=1; tmp = a[6] ; a[6] = a[5]; a[5] = tmp; swp=0; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */

if (a[0] > a[1]) { swp=1; tmp = a[1] ; a[1] = a[0]; a[0] = tmp; swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[1] > a[2]) { swp=1; tmp = a[2] ; a[2] = a[1]; a[1] = tmp; swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[2] > a[3]) { swp=1; tmp = a[3] ; a[3] = a[2]; a[2] = tmp; swp=0; }
```

```
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[3] > a[4]) { swp=1; tmp = a[4] ; a[4] = a[3]; a[3] = tmp; swp=0; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[4] > a[5]) { swp=1; tmp = a[5] ; a[5] = a[4]; a[4] = tmp; swp=0; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (a[0] > a[1]) { swp=1; tmp = a[1] ; a[1] = a[0]; a[0] = tmp; swp=0; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[1] > a[2]) { swp=1; tmp = a[2] ; a[2] = a[1]; a[1] = tmp; swp=0; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[2] > a[3]) { swp=1; tmp = a[3] ; a[3] = a[2]; a[2] = tmp; swp=0; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[3] > a[4]) { swp=1; tmp = a[4] ; a[4] = a[3]; a[3] = tmp; swp=0; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (a[0] > a[1]) { swp=1; tmp = a[1] ; a[1] = a[0]; a[0] = tmp; swp=0; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[1] > a[2]) { swp=1; tmp = a[2] ; a[2] = a[1]; a[1] = tmp; swp=0; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[2] > a[3]) { swp=1; tmp = a[3] ; a[3] = a[2]; a[2] = tmp; swp=0; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (a[0] > a[1]) { swp=1; tmp = a[1] ; a[1] = a[0]; a[0] = tmp; swp=0; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[1] > a[2]) { swp=1; tmp = a[2] ; a[2] = a[1]; a[1] = tmp; swp=0; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (a[0] > a[1]) { swp=1; tmp = a[1] ; a[1] = a[0]; a[0] = tmp; swp=0; }
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

}
```

5.5 Case D5: Swap with Goto

Here we introduce a subroutine control flow using the "goto" statement. This allows us to take the "swap" function out of line. In order to do this we have to set a variable which represents the "program counter" so the swap routine knows where to "return". This is done by setting a variable called "swp" to an integer which the swap routine can use for test and branch.

Note that the proper control flow will enter the swap branch only for these cases:

```
swp=5 /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9, } */
swp=6 /* { 2, 4, 6, 8, 1, 10, 3, 5, 7, 9, } */
swp=7 /* { 2, 4, 6, 8, 1, 3, 10, 5, 7, 9, } */
swp=8 /* { 2, 4, 6, 8, 1, 3, 5, 10, 7, 9, } */
swp=9 /* { 2, 4, 6, 8, 1, 3, 5, 7, 10, 9, } */
swp=13 /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10, } */
swp=14 /* { 2, 4, 6, 1, 8, 3, 5, 7, 9, 10, } */
swp=15 /* { 2, 4, 6, 1, 3, 8, 5, 7, 9, 10, } */
swp=16 /* { 2, 4, 6, 1, 3, 5, 8, 7, 9, 10, } */
swp=20 /* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10, } */
swp=21 /* { 2, 4, 1, 6, 3, 5, 7, 8, 9, 10, } */
swp=22 /* { 2, 4, 1, 3, 6, 5, 7, 8, 9, 10, } */
swp=26 /* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10, } */
swp=27 /* { 2, 1, 4, 3, 5, 6, 7, 8, 9, 10, } */
swp=31 /* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10, } */
```

⟨*CaseD5*⟩≡

```
#include <stdio.h>
int main()
{ int a[10] = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9};
  int tmp;
  int swp;
  int left;
  int right;
  swp = 0;
  left = 0;
  right = 0;
  /* carry the highest number to the end each time */
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[0] > a[1]) { swp=1; left=1;  right=0; goto swap; a1: swp=0; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[1] > a[2]) { swp=2; left=2;  right=1; goto swap; a2: swp=0; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[2] > a[3]) { swp=3; left=3;  right=2; goto swap; a3: swp=0; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[3] > a[4]) { swp=4; left=4;  right=3; goto swap; a4: swp=0; }
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
```

```
if (a[4] > a[5]) { swp=5; left=5;  right=4; goto swap; a5: swp=0; }
/* { 2, 4, 6, 8, 1, 10, 3, 5, 7, 9}; */
if (a[5] > a[6]) { swp=6; left=6;  right=5; goto swap; a6: swp=0; }
/* { 2, 4, 6, 8, 1, 3, 10, 5, 7, 9}; */
if (a[6] > a[7]) { swp=7; left=7;  right=6; goto swap; a7: swp=0; }
/* { 2, 4, 6, 8, 1, 3, 5, 10, 7, 9}; */
if (a[7] > a[8]) { swp=8; left=8;  right=7; goto swap; a8: swp=0; }
/* { 2, 4, 6, 8, 1, 3, 5, 7, 10, 9}; */
if (a[8] > a[9]) { swp=9; left=9;  right=8; goto swap; a9: swp=0; }
/* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */

if (a[0] > a[1]) { swp=10; left=1;  right=0; goto swap; a10: swp=0; }
/* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
if (a[1] > a[2]) { swp=11; left=2;  right=1; goto swap; a11: swp=0; }
/* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
if (a[2] > a[3]) { swp=12; left=3;  right=2; goto swap; a12: swp=0; }
/* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
if (a[3] > a[4]) { swp=13; left=4;  right=3; goto swap; a13: swp=0; }
/* { 2, 4, 6, 1, 8, 3, 5, 7, 9, 10}; */
if (a[4] > a[5]) { swp=14; left=5;  right=4; goto swap; a14: swp=0; }
/* { 2, 4, 6, 1, 3, 8, 5, 7, 9, 10}; */
if (a[5] > a[6]) { swp=15; left=6;  right=5; goto swap; a15: swp=0; }
/* { 2, 4, 6, 1, 3, 5, 8, 7, 9, 10}; */
if (a[6] > a[7]) { swp=16; left=7;  right=6; goto swap; a16: swp=0; }
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (a[7] > a[8]) { swp=17; left=8;  right=7; goto swap; a17: swp=0; }
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
/* we don't have to test the last case because a[9] is sorted */

if (a[0] > a[1]) { swp=18; left=1;  right=0; goto swap; a18: swp=0; }
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (a[1] > a[2]) { swp=19; left=2;  right=1; goto swap; a19: swp=0; }
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (a[2] > a[3]) { swp=20; left=3;  right=2; goto swap; a20: swp=0; }
/* { 2, 4, 1, 6, 3, 5, 7, 8, 9, 10}; */
if (a[3] > a[4]) { swp=21; left=4;  right=3; goto swap; a21: swp=0; }
/* { 2, 4, 1, 3, 6, 5, 7, 8, 9, 10}; */
if (a[4] > a[5]) { swp=22; left=5;  right=4; goto swap; a22: swp=0; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (a[5] > a[6]) { swp=23; left=6;  right=5; goto swap; a23: swp=0; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (a[6] > a[7]) { swp=24; left=7;  right=6; goto swap; a24: swp=0; }
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
/* a[8] and a[9] must be sorted */

if (a[0] > a[1]) { swp=25; left=1;  right=0; goto swap; a25: swp=0; }
```

```
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (a[1] > a[2]) { swp=26; left=2;  right=1; goto swap; a26: swp=0; }
/* { 2, 1, 4, 3, 5, 6, 7, 8, 9, 10}; */
if (a[2] > a[3]) { swp=27; left=3;  right=2; goto swap; a27: swp=0; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[3] > a[4]) { swp=28; left=4;  right=3; goto swap; a28: swp=0; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[4] > a[5]) { swp=29; left=5;  right=4; goto swap; a29: swp=0; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[5] > a[6]) { swp=30; left=6;  right=5; goto swap; a30: swp=0; }
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */

if (a[0] > a[1]) { swp=31; left=1;  right=0; goto swap; a31: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[1] > a[2]) { swp=32; left=2;  right=1; goto swap; a32: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[2] > a[3]) { swp=33; left=3;  right=2; goto swap; a33: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[3] > a[4]) { swp=34; left=4;  right=3; goto swap; a34: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[4] > a[5]) { swp=35; left=5;  right=4; goto swap; a35: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

if (a[0] > a[1]) { swp=36; left=1;  right=0; goto swap; a36: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[1] > a[2]) { swp=37; left=2;  right=1; goto swap; a37: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[2] > a[3]) { swp=38; left=3;  right=2; goto swap; a38: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[3] > a[4]) { swp=39; left=4;  right=3; goto swap; a39: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

if (a[0] > a[1]) { swp=40; left=1;  right=0; goto swap; a40: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[1] > a[2]) { swp=41; left=2;  right=1; goto swap; a41: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[2] > a[3]) { swp=42; left=3;  right=2; goto swap; a42: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

if (a[0] > a[1]) { swp=43; left=1;  right=0; goto swap; a43: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[1] > a[2]) { swp=44; left=2;  right=1; goto swap; a44: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

if (a[0] > a[1]) { swp=45; left=1;  right=0; goto swap; a45: swp=0; }
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
```

```
  return(0);

swap:
  tmp = a[left];
  a[left] = a[right];
  a[right] = tmp;
  if (swp ==  1) goto a1;
  if (swp ==  2) goto a2;
  if (swp ==  3) goto a3;
  if (swp ==  4) goto a4;
  if (swp ==  5) goto a5;
  if (swp ==  6) goto a6;
  if (swp ==  7) goto a7;
  if (swp ==  8) goto a8;
  if (swp ==  9) goto a9;
  if (swp == 10) goto a10;
  if (swp == 11) goto a11;
  if (swp == 12) goto a12;
  if (swp == 13) goto a13;
  if (swp == 14) goto a14;
  if (swp == 15) goto a15;
  if (swp == 16) goto a16;
  if (swp == 17) goto a17;
  if (swp == 18) goto a18;
  if (swp == 19) goto a19;
  if (swp == 20) goto a20;
  if (swp == 21) goto a21;
  if (swp == 22) goto a22;
  if (swp == 23) goto a23;
  if (swp == 24) goto a24;
  if (swp == 25) goto a25;
  if (swp == 26) goto a26;
  if (swp == 27) goto a27;
  if (swp == 28) goto a28;
  if (swp == 29) goto a29;
  if (swp == 30) goto a30;
  if (swp == 31) goto a31;
  if (swp == 32) goto a32;
  if (swp == 33) goto a33;
  if (swp == 34) goto a34;
  if (swp == 35) goto a35;
  if (swp == 36) goto a36;
  if (swp == 37) goto a37;
  if (swp == 38) goto a38;
  if (swp == 39) goto a39;
  if (swp == 40) goto a40;
```

```
  if (swp == 41) goto a41;
  if (swp == 42) goto a42;
  if (swp == 43) goto a43;
  if (swp == 44) goto a44;
  goto a45;


}
```

5.6 Case D6: Local Swap Subroutine

Here we have introduced a "proper" swap subroutine. Notice that this subroutine is in the same file as the main program so we will be able to see the XML and CCAs for this code. Later we will use library subroutines, first with a local library then with a system library.

⟨*CaseD6*⟩≡

```
#include <stdio.h>

void swap(int left, int right, int a[]) {
  int tmp;
  int i;
  tmp = a[left];
  a[left] = a[right];
  a[right] = tmp;
}


int main()
{ int a[10] = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9};
  /* carry the highest number to the end each time */
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[0] > a[1]) swap(0,1,a);
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[1] > a[2]) swap(1,2,a);
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[2] > a[3]) swap(2,3,a);
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[3] > a[4]) swap(3,4,a);
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (a[4] > a[5]) swap(4,5,a);
  /* { 2, 4, 6, 8, 1, 10, 3, 5, 7, 9}; */
  if (a[5] > a[6]) swap(5,6,a);
  /* { 2, 4, 6, 8, 1, 3, 10, 5, 7, 9}; */
  if (a[6] > a[7]) swap(6,7,a);
  /* { 2, 4, 6, 8, 1, 3, 5, 10, 7, 9}; */
  if (a[7] > a[8]) swap(7,8,a);
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 10, 9}; */
  if (a[8] > a[9]) swap(8,9,a);
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */

  if (a[0] > a[1]) swap(0,1,a);
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
  if (a[1] > a[2]) swap(1,2,a);
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
  if (a[2] > a[3]) swap(2,3,a);
```

```
/* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
if (a[3] > a[4]) swap(3,4,a);
/* { 2, 4, 6, 1, 8, 3, 5, 7, 9, 10}; */
if (a[4] > a[5]) swap(4,5,a);
/* { 2, 4, 6, 1, 3, 8, 5, 7, 9, 10}; */
if (a[5] > a[6]) swap(5,6,a);
/* { 2, 4, 6, 1, 3, 5, 8, 7, 9, 10}; */
if (a[6] > a[7]) swap(6,7,a);
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (a[7] > a[8]) swap(7,8,a);
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
/* we don't have to test the last case because a[9] is sorted */

if (a[0] > a[1]) swap(0,1,a);
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (a[1] > a[2]) swap(1,2,a);
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (a[2] > a[3]) swap(2,3,a);
/* { 2, 4, 1, 6, 3, 5, 7, 8, 9, 10}; */
if (a[3] > a[4]) swap(3,4,a);
/* { 2, 4, 1, 3, 6, 5, 7, 8, 9, 10}; */
if (a[4] > a[5]) swap(4,5,a);
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (a[5] > a[6]) swap(5,6,a);
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (a[6] > a[7]) swap(6,7,a);
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
/* a[8] and a[9] must be sorted */

if (a[0] > a[1]) swap(0,1,a);
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (a[1] > a[2]) swap(1,2,a);
/* { 2, 1, 4, 3, 5, 6, 7, 8, 9, 10}; */
if (a[2] > a[3]) swap(2,3,a);
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[3] > a[4]) swap(3,4,a);
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[4] > a[5]) swap(4,5,a);
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[5] > a[6]) swap(5,6,a);
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */

if (a[0] > a[1]) swap(0,1,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (a[1] > a[2]) swap(1,2,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
```

```
  if (a[2] > a[3]) swap(2,3,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[3] > a[4]) swap(3,4,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[4] > a[5]) swap(4,5,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (a[0] > a[1]) swap(0,1,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[1] > a[2]) swap(1,2,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[2] > a[3]) swap(2,3,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[3] > a[4]) swap(3,4,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (a[0] > a[1]) swap(0,1,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[1] > a[2]) swap(1,2,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[2] > a[3]) swap(2,3,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (a[0] > a[1]) swap(0,1,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (a[1] > a[2]) swap(1,2,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (a[0] > a[1]) swap(0,1,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  return(0);


}
```

5.7 Case D7: Local Compare and Swap Subroutine

Here we have buried the control flow logic into a subroutine just to see how this changes the CCAs.

If the control flow is correct then these are the only compares that should succeed and the only swaps that need to happen:

```
cmp(4,5) ==> swap(4,5) 2 4 6 8 10 1 3 5 7 9
cmp(5,6) ==> swap(5,6) 2 4 6 8 1 10 3 5 7 9
cmp(6,7) ==> swap(6,7) 2 4 6 8 1 3 10 5 7 9
cmp(7,8) ==> swap(7,8) 2 4 6 8 1 3 5 10 7 9
cmp(8,9) ==> swap(8,9) 2 4 6 8 1 3 5 7 10 9
cmp(3,4) ==> swap(3,4) 2 4 6 8 1 3 5 7 9 10
cmp(4,5) ==> swap(4,5) 2 4 6 1 8 3 5 7 9 10
cmp(5,6) ==> swap(5,6) 2 4 6 1 3 8 5 7 9 10
cmp(6,7) ==> swap(6,7) 2 4 6 1 3 5 8 7 9 10
cmp(2,3) ==> swap(2,3) 2 4 6 1 3 5 7 8 9 10
cmp(3,4) ==> swap(3,4) 2 4 1 6 3 5 7 8 9 10
cmp(4,5) ==> swap(4,5) 2 4 1 3 6 5 7 8 9 10
cmp(1,2) ==> swap(1,2) 2 4 1 3 5 6 7 8 9 10
cmp(2,3) ==> swap(2,3) 2 1 4 3 5 6 7 8 9 10
cmp(0,1) ==> swap(0,1) 2 1 3 4 5 6 7 8 9 10
```

⟨*CaseD7*⟩≡

```
#include <stdio.h>

int compare(int left, int right, int a[]) {
  int retval=0;
  if (a[left] > a[right]) retval=1;
  return(retval);
}

void swap(int left, int right, int a[]) {
  int tmp;
  tmp = a[left];
  a[left] = a[right];
  a[right] = tmp;
}


int main()
{ int a[10] = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9};
  /* carry the highest number to the end each time */
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (compare(0,1,a) == 1) swap(0,1,a);
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (compare(1,2,a) == 1) swap(1,2,a);
```

```
/* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
if (compare(2,3,a) == 1) swap(2,3,a);
/* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
if (compare(3,4,a) == 1) swap(3,4,a);
/* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
if (compare(4,5,a) == 1) swap(4,5,a);
/* { 2, 4, 6, 8, 1, 10, 3, 5, 7, 9}; */
if (compare(5,6,a) == 1) swap(5,6,a);
/* { 2, 4, 6, 8, 1, 3, 10, 5, 7, 9}; */
if (compare(6,7,a) == 1) swap(6,7,a);
/* { 2, 4, 6, 8, 1, 3, 5, 10, 7, 9}; */
if (compare(7,8,a) == 1) swap(7,8,a);
/* { 2, 4, 6, 8, 1, 3, 5, 7, 10, 9}; */
if (compare(8,9,a) == 1) swap(8,9,a);
/* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */

if (compare(0,1,a) == 1) swap(0,1,a);
/* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
if (compare(1,2,a) == 1) swap(1,2,a);
/* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
if (compare(2,3,a) == 1) swap(2,3,a);
/* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
if (compare(3,4,a) == 1) swap(3,4,a);
/* { 2, 4, 6, 1, 8, 3, 5, 7, 9, 10}; */
if (compare(4,5,a) == 1) swap(4,5,a);
/* { 2, 4, 6, 1, 3, 8, 5, 7, 9, 10}; */
if (compare(5,6,a) == 1) swap(5,6,a);
/* { 2, 4, 6, 1, 3, 5, 8, 7, 9, 10}; */
if (compare(6,7,a) == 1) swap(6,7,a);
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (compare(7,8,a) == 1) swap(7,8,a);
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
/* we don't have to test the last case because a[9] is sorted */

if (compare(0,1,a) == 1) swap(0,1,a);
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (compare(1,2,a) == 1) swap(1,2,a);
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (compare(2,3,a) == 1) swap(2,3,a);
/* { 2, 4, 1, 6, 3, 5, 7, 8, 9, 10}; */
if (compare(3,4,a) == 1) swap(3,4,a);
/* { 2, 4, 1, 3, 6, 5, 7, 8, 9, 10}; */
if (compare(4,5,a) == 1) swap(4,5,a);
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (compare(5,6,a) == 1) swap(5,6,a);
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
```

```
if (compare(6,7,a) == 1) swap(6,7,a);
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
/* a[8] and a[9] must be sorted */

if (compare(0,1,a) == 1) swap(0,1,a);
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (compare(1,2,a) == 1) swap(1,2,a);
/* { 2, 1, 4, 3, 5, 6, 7, 8, 9, 10}; */
if (compare(2,3,a) == 1) swap(2,3,a);
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(3,4,a) == 1) swap(3,4,a);
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(4,5,a) == 1) swap(4,5,a);
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(5,6,a) == 1) swap(5,6,a);
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */

if (compare(0,1,a) == 1) swap(0,1,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(1,2,a) == 1) swap(1,2,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(2,3,a) == 1) swap(2,3,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(3,4,a) == 1) swap(3,4,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(4,5,a) == 1) swap(4,5,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

if (compare(0,1,a) == 1) swap(0,1,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(1,2,a) == 1) swap(1,2,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(2,3,a) == 1) swap(2,3,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(3,4,a) == 1) swap(3,4,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

if (compare(0,1,a) == 1) swap(0,1,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(1,2,a) == 1) swap(1,2,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(2,3,a) == 1) swap(2,3,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

if (compare(0,1,a) == 1) swap(0,1,a);
/* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
```

```
  if (compare(1,2,a) == 1) swap(1,2,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (compare(0,1,a) == 1) swap(0,1,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  return(0);


}
```

5.8 Case D8: Local String Compare

Now we have changed the data shape considerably. Here we are doing the compare as before except that the data represents strings which in this case are 2 bytes long (one for the character and one for the null).

⟨*CaseD8*⟩≡

```
#include <stdio.h>

int compare(int left, int right, char **a) {
  int retval=0;
  if (*a[left] > *a[right]) retval=1;
  return(retval);
}

void swap(int left, int right, char **a) {
  char *tmp;
  tmp = a[left];
  a[left] = a[right];
  a[right] = tmp;
}

int main()
{ char *a[10] = { "b", "d", "f", "h", "j", "a", "c", "e", "g", "i"};
  char *tmp;
  int i;
  /* carry the highest number to the end each time */
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (compare(0,1,a) == 1) swap(0,1,a);
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (compare(1,2,a) == 1) swap(1,2,a);
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (compare(2,3,a) == 1) swap(2,3,a);
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (compare(3,4,a) == 1) swap(3,4,a);
  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
  if (compare(4,5,a) == 1) swap(4,5,a);
  /* { 2, 4, 6, 8, 1, 10, 3, 5, 7, 9}; */
  if (compare(5,6,a) == 1) swap(5,6,a);
  /* { 2, 4, 6, 8, 1, 3, 10, 5, 7, 9}; */
  if (compare(6,7,a) == 1) swap(6,7,a);
  /* { 2, 4, 6, 8, 1, 3, 5, 10, 7, 9}; */
  if (compare(7,8,a) == 1) swap(7,8,a);
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 10, 9}; */
  if (compare(8,9,a) == 1) swap(8,9,a);
  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
```

```
if (compare(0,1,a) == 1) swap(0,1,a);
/* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
if (compare(1,2,a) == 1) swap(1,2,a);
/* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
if (compare(2,3,a) == 1) swap(2,3,a);
/* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
if (compare(3,4,a) == 1) swap(3,4,a);
/* { 2, 4, 6, 1, 8, 3, 5, 7, 9, 10}; */
if (compare(4,5,a) == 1) swap(4,5,a);
/* { 2, 4, 6, 1, 3, 8, 5, 7, 9, 10}; */
if (compare(5,6,a) == 1) swap(5,6,a);
/* { 2, 4, 6, 1, 3, 5, 8, 7, 9, 10}; */
if (compare(6,7,a) == 1) swap(6,7,a);
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (compare(7,8,a) == 1) swap(7,8,a);
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
/* we don't have to test the last case because a[9] is sorted */

if (compare(0,1,a) == 1) swap(0,1,a);
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (compare(1,2,a) == 1) swap(1,2,a);
/* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
if (compare(2,3,a) == 1) swap(2,3,a);
/* { 2, 4, 1, 6, 3, 5, 7, 8, 9, 10}; */
if (compare(3,4,a) == 1) swap(3,4,a);
/* { 2, 4, 1, 3, 6, 5, 7, 8, 9, 10}; */
if (compare(4,5,a) == 1) swap(4,5,a);
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (compare(5,6,a) == 1) swap(5,6,a);
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (compare(6,7,a) == 1) swap(6,7,a);
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
/* a[8] and a[9] must be sorted */

if (compare(0,1,a) == 1) swap(0,1,a);
/* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
if (compare(1,2,a) == 1) swap(1,2,a);
/* { 2, 1, 4, 3, 5, 6, 7, 8, 9, 10}; */
if (compare(2,3,a) == 1) swap(2,3,a);
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(3,4,a) == 1) swap(3,4,a);
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(4,5,a) == 1) swap(4,5,a);
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
if (compare(5,6,a) == 1) swap(5,6,a);
/* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
```

```
  if (compare(0,1,a) == 1) swap(0,1,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (compare(1,2,a) == 1) swap(1,2,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (compare(2,3,a) == 1) swap(2,3,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (compare(3,4,a) == 1) swap(3,4,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (compare(4,5,a) == 1) swap(4,5,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (compare(0,1,a) == 1) swap(0,1,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (compare(1,2,a) == 1) swap(1,2,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (compare(2,3,a) == 1) swap(2,3,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (compare(3,4,a) == 1) swap(3,4,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (compare(0,1,a) == 1) swap(0,1,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (compare(1,2,a) == 1) swap(1,2,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (compare(2,3,a) == 1) swap(2,3,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (compare(0,1,a) == 1) swap(0,1,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  if (compare(1,2,a) == 1) swap(1,2,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */

  if (compare(0,1,a) == 1) swap(0,1,a);
  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
  return(0);

}
```

6 E: Change of System Libraries

6.1 Case E1: Local Subroutine Library, static link

Here we pick up the compare and the swap subroutine, put them into a library, and statically link the library code.

⟨*CaseE1*⟩≡

6.2 Case E2: Local Subroutine Library, dynamic link

Here we pick up the compare and the swap subroutine, put them into a library, and dynamically link the library code.

⟨*CaseE2*⟩≡

6.3 Case E3: Strlen

Here we use the system strlen subroutine to compute a property of the data without changing the data.

⟨*CaseE3*⟩≡

6.4 Case E4: Strcpy

Here we use a system strcpy subroutine to change the data

⟨*CaseE4*⟩≡

7 F: Change of System Input

We have been avoiding input and output issues. Now is the time to introduce them as a few people insist on using input and output.

7.1 Case F1: 10 Item Command Line

Here we collect the 10 items from the command line.

⟨*CaseF1*⟩≡

7.2 Case F2: 10 Item Read Line

Here we collect the 10 items from the read function.

⟨*CaseF2*⟩≡

8 G: Change of Repository Features

Here we compile the linear bubble sort with many different compiler switches so we can do comparisons of the extractions and the CCAs. The repository should be able to tell us which compiler switches were used if we store that information with the binary.

8.1 Case G1: Compiler Switches

⟨*CaseG1*⟩≡

8.2 Case G2: Multiple C Compilers

Here we compile the linear bubble sort with an assortment of C compilers. This will test how the repository handles many different kinds of compiler output.

⟨*CaseG2*⟩≡

8.3 Case G3: Assembler

Here we have an assembler program which is equivalent to the linear bubble sort routine. This will test our ability to handle different languages.

⟨*CaseG3*⟩≡

```
push    ebp
mov     ebp,esp
sub     esp,BYTE 8
call    DWORD $+86
call    DWORD $+177
call    DWORD $+1532
leave
ret
push    DWORD [134519160]
jmp     DWORD [134519164]
add     BYTE [eax],al
add     BYTE [eax],al
jmp     DWORD [134519168]
push    DWORD 0x00000000
jmp     NEAR $-0x0000001B
xor     ebp,ebp
pop     esi
mov     ecx,esp
and     esp,BYTE 240
push    eax
push    esp
push    edx
push    DWORD 0x080487E4
push    DWORD 0x080487B4
push    ecx
push    esi
push    DWORD 0x080482F4
call    DWORD $-44
hlt
nop
nop
push    ebp
mov     ebp,esp
push    ebx
push    eax
call    DWORD $+5
pop     ebx
add     ebx,DWORD 5890
mov     eax,DWORD [ebx+16]
test    eax,eax
je      SHORT $+4
```

```
        call       eax
        mov        ebx,DWORD [ebp-4]
        leave
        ret
        nop
        nop
        push       ebp
        mov        ebp,esp
        sub        esp,BYTE 8
        cmp        BYTE [134519176],0
        jne        SHORT $+0x2B
        mov        eax,DWORD [134518932]
        mov        edx,DWORD [eax]
        test       edx,edx
        je         SHORT $+0x19
        mov        esi,esi
        add        eax,BYTE 4
        mov        DWORD [ds:134518932],eax
        call       edx
        mov        eax,DWORD [134518932]
        mov        edx,DWORD [eax]
        test       edx,edx
        jne        SHORT $-0x13
        mov        BYTE [134519176],1
        leave
        ret
        mov        esi,esi
        push       ebp
        mov        ebp,esp
        sub        esp,BYTE 8
        mov        eax,DWORD [134519152]
        test       eax,eax
        je         SHORT $+0x1B
        mov        eax,DWORD 0x00000000
        test       eax,eax
        je         SHORT $+0x12
        sub        esp,BYTE 12
        push       DWORD 0x08049970
        call       DWORD $-134513384
        add        esp,BYTE 16
        leave
        ret
        nop
        nop
; int a[10] = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9};
        push       ebp
```

```
        mov     ebp,esp
        push    edi
        push    esi
        sub     esp,BYTE 64
        and     esp,BYTE 240
        mov     eax,DWORD 0x00000000
        sub     esp,eax
        lea     edi,[ebp-56]
        mov     esi,DWORD 0x08048860
        cld
        mov     eax,DWORD 0x0000000A
        mov     ecx,eax
repz    movsd
        mov     eax,DWORD [ebp-56]
;  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
;  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
        cmp     eax,DWORD [ebp-52]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-52]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-56]
        mov     DWORD [ebp-52],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-56],eax
        mov     eax,DWORD [ebp-52]
;  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
;  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
        cmp     eax,DWORD [ebp-48]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-48]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-52]
        mov     DWORD [ebp-48],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-52],eax
        mov     eax,DWORD [ebp-48]
;  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
;  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
        cmp     eax,DWORD [ebp-44]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-44]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-48]
        mov     DWORD [ebp-44],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-48],eax
```

```
        mov     eax,DWORD [ebp-44]
;  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
;  /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
        cmp     eax,DWORD [ebp-40]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-40]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-44]
        mov     DWORD [ebp-40],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-44],eax
        mov     eax,DWORD [ebp-40]
;  if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
;  /* { 2, 4, 6, 8, 1, 10, 3, 5, 7, 9}; */
        cmp     eax,DWORD [ebp-36]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-36]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-40]
        mov     DWORD [ebp-36],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-40],eax
        mov     eax,DWORD [ebp-36]
;  if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
;  /* { 2, 4, 6, 8, 1, 3, 10, 5, 7, 9}; */
        cmp     eax,DWORD [ebp-32]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-32]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-36]
        mov     DWORD [ebp-32],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-36],eax
        mov     eax,DWORD [ebp-32]
;  if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
;  /* { 2, 4, 6, 8, 1, 3, 5, 10, 7, 9}; */
        cmp     eax,DWORD [ebp-28]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-28]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-32]
        mov     DWORD [ebp-28],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-32],eax
        mov     eax,DWORD [ebp-28]
;  if (a[7] > a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
```

```
;  /* { 2, 4, 6, 8, 1, 3, 5, 7, 10, 9}; */
        cmp         eax,DWORD [ebp-24]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-24]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-28]
        mov         DWORD [ebp-24],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-28],eax
        mov         eax,DWORD [ebp-24]
;  if (a[8] > a[9]) { tmp = a[9] ; a[9] = a[8]; a[8] = tmp; }
;  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
        cmp         eax,DWORD [ebp-20]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-20]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-24]
        mov         DWORD [ebp-20],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-24],eax
        mov         eax,DWORD [ebp-56]
;  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
;  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
        cmp         eax,DWORD [ebp-52]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-52]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-56]
        mov         DWORD [ebp-52],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-56],eax
        mov         eax,DWORD [ebp-52]
;  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
;  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
        cmp         eax,DWORD [ebp-48]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-48]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-52]
        mov         DWORD [ebp-48],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-52],eax
        mov         eax,DWORD [ebp-48]
;  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
;  /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
        cmp         eax,DWORD [ebp-44]
```

```
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-44]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-48]
        mov         DWORD [ebp-44],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-48],eax
        mov         eax,DWORD [ebp-44]
;  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
;  /* { 2, 4, 6, 1, 8, 3, 5, 7, 9, 10}; */
        cmp         eax,DWORD [ebp-40]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-40]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-44]
        mov         DWORD [ebp-40],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-44],eax
        mov         eax,DWORD [ebp-40]
;  if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
;  /* { 2, 4, 6, 1, 3, 8, 5, 7, 9, 10}; */
        cmp         eax,DWORD [ebp-36]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-36]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-40]
        mov         DWORD [ebp-36],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-40],eax
        mov         eax,DWORD [ebp-36]
;  if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
;  /* { 2, 4, 6, 1, 3, 5, 8, 7, 9, 10}; */
        cmp         eax,DWORD [ebp-32]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-32]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-36]
        mov         DWORD [ebp-32],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-36],eax
        mov         eax,DWORD [ebp-32]
;  if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
;  /* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
        cmp         eax,DWORD [ebp-28]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-28]
```

```
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-32]
        mov     DWORD [ebp-28],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-32],eax
        mov     eax,DWORD [ebp-28]
;  if (a[7] > a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
;  /* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
        cmp     eax,DWORD [ebp-24]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-24]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-28]
        mov     DWORD [ebp-24],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-28],eax
        mov     eax,DWORD [ebp-56]
;  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
;  /* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
        cmp     eax,DWORD [ebp-52]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-52]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-56]
        mov     DWORD [ebp-52],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-56],eax
        mov     eax,DWORD [ebp-52]
;  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
;  /* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
        cmp     eax,DWORD [ebp-48]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-48]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-52]
        mov     DWORD [ebp-48],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-52],eax
        mov     eax,DWORD [ebp-48]
;  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
;  /* { 2, 4, 1, 6, 3, 5, 7, 8, 9, 10}; */
        cmp     eax,DWORD [ebp-44]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-44]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-48]
```

```
        mov         DWORD [ebp-44],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-48],eax
        mov         eax,DWORD [ebp-44]
;  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
;  /* { 2, 4, 1, 3, 6, 5, 7, 8, 9, 10}; */
        cmp         eax,DWORD [ebp-40]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-40]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-44]
        mov         DWORD [ebp-40],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-44],eax
        mov         eax,DWORD [ebp-40]
;  if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
;  /* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
        cmp         eax,DWORD [ebp-36]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-36]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-40]
        mov         DWORD [ebp-36],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-40],eax
        mov         eax,DWORD [ebp-36]
;  if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
;  /* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
        cmp         eax,DWORD [ebp-32]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-32]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-36]
        mov         DWORD [ebp-32],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-36],eax
        mov         eax,DWORD [ebp-32]
;  if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
;  /* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
        cmp         eax,DWORD [ebp-28]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-28]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-32]
        mov         DWORD [ebp-28],eax
        mov         eax,DWORD [ebp-60]
```

```
        mov        DWORD [ebp-32],eax
        mov        eax,DWORD [ebp-56]
;  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
;  /* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
        cmp        eax,DWORD [ebp-52]
        jle        SHORT $+0x14
        mov        eax,DWORD [ebp-52]
        mov        DWORD [ebp-60],eax
        mov        eax,DWORD [ebp-56]
        mov        DWORD [ebp-52],eax
        mov        eax,DWORD [ebp-60]
        mov        DWORD [ebp-56],eax
        mov        eax,DWORD [ebp-52]
;  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
;  /* { 2, 1, 4, 3, 5, 6, 7, 8, 9, 10}; */
        cmp        eax,DWORD [ebp-48]
        jle        SHORT $+0x14
        mov        eax,DWORD [ebp-48]
        mov        DWORD [ebp-60],eax
        mov        eax,DWORD [ebp-52]
        mov        DWORD [ebp-48],eax
        mov        eax,DWORD [ebp-60]
        mov        DWORD [ebp-52],eax
        mov        eax,DWORD [ebp-48]
;  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
;  /* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
        cmp        eax,DWORD [ebp-44]
        jle        SHORT $+0x14
        mov        eax,DWORD [ebp-44]
        mov        DWORD [ebp-60],eax
        mov        eax,DWORD [ebp-48]
        mov        DWORD [ebp-44],eax
        mov        eax,DWORD [ebp-60]
        mov        DWORD [ebp-48],eax
        mov        eax,DWORD [ebp-44]
;  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
;  /* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
        cmp        eax,DWORD [ebp-40]
        jle        SHORT $+0x14
        mov        eax,DWORD [ebp-40]
        mov        DWORD [ebp-60],eax
        mov        eax,DWORD [ebp-44]
        mov        DWORD [ebp-40],eax
        mov        eax,DWORD [ebp-60]
        mov        DWORD [ebp-44],eax
        mov        eax,DWORD [ebp-40]
```

```
;  if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
;  /* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
       cmp        eax,DWORD [ebp-36]
       jle        SHORT $+0x14
       mov        eax,DWORD [ebp-36]
       mov        DWORD [ebp-60],eax
       mov        eax,DWORD [ebp-40]
       mov        DWORD [ebp-36],eax
       mov        eax,DWORD [ebp-60]
       mov        DWORD [ebp-40],eax
       mov        eax,DWORD [ebp-36]
;  if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
;  /* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
       cmp        eax,DWORD [ebp-32]
       jle        SHORT $+0x14
       mov        eax,DWORD [ebp-32]
       mov        DWORD [ebp-60],eax
       mov        eax,DWORD [ebp-36]
       mov        DWORD [ebp-32],eax
       mov        eax,DWORD [ebp-60]
       mov        DWORD [ebp-36],eax
       mov        eax,DWORD [ebp-56]
;  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
       cmp        eax,DWORD [ebp-52]
       jle        SHORT $+0x14
       mov        eax,DWORD [ebp-52]
       mov        DWORD [ebp-60],eax
       mov        eax,DWORD [ebp-56]
       mov        DWORD [ebp-52],eax
       mov        eax,DWORD [ebp-60]
       mov        DWORD [ebp-56],eax
       mov        eax,DWORD [ebp-52]
;  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
       cmp        eax,DWORD [ebp-48]
       jle        SHORT $+0x14
       mov        eax,DWORD [ebp-48]
       mov        DWORD [ebp-60],eax
       mov        eax,DWORD [ebp-52]
       mov        DWORD [ebp-48],eax
       mov        eax,DWORD [ebp-60]
       mov        DWORD [ebp-52],eax
       mov        eax,DWORD [ebp-48]
;  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
```

```
        cmp         eax,DWORD [ebp-44]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-44]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-48]
        mov         DWORD [ebp-44],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-48],eax
        mov         eax,DWORD [ebp-44]
;  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
        cmp         eax,DWORD [ebp-40]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-40]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-44]
        mov         DWORD [ebp-40],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-44],eax
        mov         eax,DWORD [ebp-40]
;  if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
        cmp         eax,DWORD [ebp-36]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-36]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-40]
        mov         DWORD [ebp-36],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-40],eax
        mov         eax,DWORD [ebp-56]
;  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
        cmp         eax,DWORD [ebp-52]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-52]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-56]
        mov         DWORD [ebp-52],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-56],eax
        mov         eax,DWORD [ebp-52]
;  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
        cmp         eax,DWORD [ebp-48]
        jle         SHORT $+0x14
```

```
        mov     eax,DWORD [ebp-48]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-52]
        mov     DWORD [ebp-48],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-52],eax
        mov     eax,DWORD [ebp-48]
;  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
        cmp     eax,DWORD [ebp-44]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-44]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-48]
        mov     DWORD [ebp-44],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-48],eax
        mov     eax,DWORD [ebp-44]
;  if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
        cmp     eax,DWORD [ebp-40]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-40]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-44]
        mov     DWORD [ebp-40],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-44],eax
        mov     eax,DWORD [ebp-56]
;  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
        cmp     eax,DWORD [ebp-52]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-52]
        mov     DWORD [ebp-60],eax
        mov     eax,DWORD [ebp-56]
        mov     DWORD [ebp-52],eax
        mov     eax,DWORD [ebp-60]
        mov     DWORD [ebp-56],eax
        mov     eax,DWORD [ebp-52]
;  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
        cmp     eax,DWORD [ebp-48]
        jle     SHORT $+0x14
        mov     eax,DWORD [ebp-48]
        mov     DWORD [ebp-60],eax
```

```
        mov         eax,DWORD [ebp-52]
        mov         DWORD [ebp-48],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-52],eax
        mov         eax,DWORD [ebp-48]
;  if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
        cmp         eax,DWORD [ebp-44]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-44]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-48]
        mov         DWORD [ebp-44],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-48],eax
        mov         eax,DWORD [ebp-56]
;  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
        cmp         eax,DWORD [ebp-52]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-52]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-56]
        mov         DWORD [ebp-52],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-56],eax
        mov         eax,DWORD [ebp-52]
;  if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
        cmp         eax,DWORD [ebp-48]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-48]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-52]
        mov         DWORD [ebp-48],eax
        mov         eax,DWORD [ebp-60]
        mov         DWORD [ebp-52],eax
        mov         eax,DWORD [ebp-56]
;  if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
;  /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
        cmp         eax,DWORD [ebp-52]
        jle         SHORT $+0x14
        mov         eax,DWORD [ebp-52]
        mov         DWORD [ebp-60],eax
        mov         eax,DWORD [ebp-56]
        mov         DWORD [ebp-52],eax
```

```
mov         eax,DWORD [ebp-60]
mov         DWORD [ebp-56],eax
lea         esp,[ebp-8]
pop         esi
pop         edi
leave
ret
nop
nop
nop
push        ebp
mov         ebp,esp
push        esi
push        ebx
call        DWORD $-1453
mov         eax,DWORD 0x0804988C
sub         eax,DWORD 0x0804988C
sar         eax,BYTE 2
xor         ebx,ebx
cmp         ebx,eax
jae         SHORT $+0x11
mov         esi,eax
nop
call        DWORD [ebp+ebx*4+134518924]
inc         ebx
cmp         ebx,esi
jb          SHORT $-0xA
pop         ebx
pop         esi
leave
ret
push        ebp
mov         ebp,esp
push        ebx
push        eax
mov         eax,DWORD 0x0804988C
sub         eax,DWORD 0x0804988C
sar         eax,BYTE 2
test        eax,eax
lea         ebx,[eax-1]
jne         SHORT $+0xD
mov         ebx,DWORD [ebp-4]
leave
jmp         NEAR $+0x0000003B
mov         esi,esi
call        DWORD [ebp+ebx*4+134518924]
```

```
mov        edx,ebx
dec        ebx
test       edx,edx
jne        SHORT $-0xC
jmp        SHORT $-0x19
push       ebp
mov        ebp,esp
push       ebx
push       edx
mov        eax,DWORD [134519136]
cmp        eax,BYTE 255
mov        ebx,DWORD 0x08049960
je         SHORT $+0xE
sub        ebx,BYTE 4
call       eax
mov        eax,DWORD [ebx]
cmp        eax,BYTE 255
jne        SHORT $-0xA
pop        eax
pop        ebx
leave
ret
push       ebp
mov        ebp,esp
push       ebx
push       edx
call       DWORD $+5
pop        ebx
add        ebx,DWORD 4398
call       DWORD $-1473
mov        ebx,DWORD [ebp-4]
leave
ret
```

8.4 Case G4: Fortran

Here we have a fortran program which is equivalent to the linear bubble sort routine. This will test our ability to handle different languages.

⟨*CaseG4*⟩≡

```
      C int main()
      C { int a[10] = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9};
      C   int tmp;
      C   /* carry the highest number to the end each time */
      C   /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
      C   if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
      C   /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
        DIMENSION I[10]
        I[0] = 2
        I[1] = 4
        I[2] = 6
        I[3] = 8
        I[4] = 10
        I[5] = 1
        I[6] = 3
        I[7] = 5
        I[8] = 7
        I[9] = 9
        IF (I[0] .LE. I[1]) GOTO A1
        J = I[1]
        I[1] = I[0]
        I[0] = J
      C   if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
      C   /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
 1     IF (I[1] .LE. I[2]) GOTO 2
       J = I[2]
       I[2] = I[1]
       I[1] = J
      C   if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
      C   /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
 2     IF (I[2] .LE. I[3]) GOTO 3
       J = I[3]
       I[3] = I[2]
       I[2] = J
      C   if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
      C   /* { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9}; */
 3     IF (I[3] .LE. I[4]) GOTO 4
       J = I[4]
       I[4] = I[3]
       I[3] = J
      C   if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
```

```
      C   /* { 2, 4, 6, 8, 1, 10, 3, 5, 7, 9}; */
4      IF (I[4] .LE. I[5]) GOTO 5
       J = I[5]
       I[5] = I[4]
       I[4] = J
      C   if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
      C   /* { 2, 4, 6, 8, 1, 3, 10, 5, 7, 9}; */
5      IF (I[5] .LE. I[6]) GOTO 6
       J = I[6]
       I[6] = I[5]
       I[5] = J
      C   if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
      C   /* { 2, 4, 6, 8, 1, 3, 5, 10, 7, 9}; */
6      IF (I[6] .LE. I[7]) GOTO 7
       J = I[7]
       I[7] = I[6]
       I[6] = J
      C   if (a[7] > a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
      C   /* { 2, 4, 6, 8, 1, 3, 5, 7, 10, 9}; */
7      IF (I[7] .LE. I[8]) GOTO 8
       J = I[8]
       I[8] = I[7]
       I[7] = J
      C   if (a[8] > a[9]) { tmp = a[9] ; a[9] = a[8]; a[8] = tmp; }
      C   /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
8      IF (I[8] .LE. I[9]) GOTO 9
       J = I[9]
       I[9] = I[8]
       I[8] = J
      C
      C   if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
      C   /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
9      IF (I[0] .LE. I[1]) GOTO 10
       J = I[1]
       I[1] = I[0]
       I[0] = J
      C   if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
      C   /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
10     IF (I[1] .LE. I[2]) GOTO 11
       J = I[2]
       I[2] = I[1]
       I[1] = J
      C   if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
      C   /* { 2, 4, 6, 8, 1, 3, 5, 7, 9, 10}; */
11     IF (I[2] .LE. I[3]) GOTO 12
       J = I[3]
```

```
       I[3] = I[2]
       I[2] = J
      C   if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
      C   /* { 2, 4, 6, 1, 8, 3, 5, 7, 9, 10}; */
12     IF (I[3] .LE. I[4]) GOTO 13
       J = I[4]
       I[4] = I[3]
       I[3] = J
      C   if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
      C   /* { 2, 4, 6, 1, 3, 8, 5, 7, 9, 10}; */
13     IF (I[4] .LE. I[5]) GOTO 14
       J = I[5]
       I[5] = I[4]
       I[4] = J
      C   if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
      C   /* { 2, 4, 6, 1, 3, 5, 8, 7, 9, 10}; */
14     IF (I[5] .LE. I[6]) GOTO 15
       J = I[6]
       I[6] = I[5]
       I[5] = J
      C   if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
      C   /* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
15     IF (I[6] .LE. I[7]) GOTO 16
       J = I[7]
       I[7] = I[6]
       I[6] = J
      C   if (a[7] > a[8]) { tmp = a[8] ; a[8] = a[7]; a[7] = tmp; }
      C   /* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
16     IF (I[7] .LE. I[8]) GOTO 17
       J = I[8]
       I[8] = I[7]
       I[7] = J
      C   /* we don't have to test the last case because a[9] is sorted */
      C
      C   if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
      C   /* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
17     IF (I[0] .LE. I[1]) GOTO 18
       J = I[1]
       I[1] = I[0]
       I[0] = J
      C   if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
      C   /* { 2, 4, 6, 1, 3, 5, 7, 8, 9, 10}; */
18     IF (I[1] .LE. I[2]) GOTO 19
       J = I[2]
       I[2] = I[1]
       I[1] = J
```

```
      C   if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
      C   /* { 2, 4, 1, 6, 3, 5, 7, 8, 9, 10}; */
19     IF (I[2] .LE. I[3]) GOTO 20
       J = I[3]
       I[3] = I[2]
       I[2] = J
      C   if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
      C   /* { 2, 4, 1, 3, 6, 5, 7, 8, 9, 10}; */
20     IF (I[3] .LE. I[4]) GOTO 21
       J = I[4]
       I[4] = I[3]
       I[3] = J
      C   if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
      C   /* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
21     IF (I[4] .LE. I[5]) GOTO 22
       J = I[5]
       I[5] = I[4]
       I[4] = J
      C   if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
      C   /* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
22     IF (I[5] .LE. I[6]) GOTO 23
       J = I[6]
       I[6] = I[5]
       I[5] = J
      C   if (a[6] > a[7]) { tmp = a[7] ; a[7] = a[6]; a[6] = tmp; }
      C   /* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
23     IF (I[6] .LE. I[7]) GOTO 24
       J = I[7]
       I[7] = I[6]
       I[6] = J
      C   /* a[8] and a[9] must be sorted */
      C
      C   if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
      C   /* { 2, 4, 1, 3, 5, 6, 7, 8, 9, 10}; */
24     IF (I[0] .LE. I[1]) GOTO 25
       J = I[1]
       I[1] = I[0]
       I[0] = J
      C   if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
      C   /* { 2, 1, 4, 3, 5, 6, 7, 8, 9, 10}; */
25     IF (I[1] .LE. I[2]) GOTO 26
       J = I[2]
       I[2] = I[1]
       I[1] = J
      C   if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
      C   /* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
```

```
26     IF (I[2] .LE. I[3]) GOTO 27
       J = I[3]
       I[3] = I[2]
       I[2] = J
      C   if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
      C   /* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
27     IF (I[3] .LE. I[4]) GOTO 28
       J = I[4]
       I[4] = I[3]
       I[3] = J
      C   if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
      C   /* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
28     IF (I[4] .LE. I[5]) GOTO 29
       J = I[5]
       I[5] = I[4]
       I[4] = J
      C   if (a[5] > a[6]) { tmp = a[6] ; a[6] = a[5]; a[5] = tmp; }
      C   /* { 2, 1, 3, 4, 5, 6, 7, 8, 9, 10}; */
29     IF (I[5] .LE. I[6]) GOTO 30
       J = I[6]
       I[6] = I[5]
       I[5] = J
      C
      C   if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
30     IF (I[0] .LE. I[1]) GOTO 31
       J = I[1]
       I[1] = I[0]
       I[0] = J
      C   if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
31     IF (I[1] .LE. I[2]) GOTO 32
       J = I[2]
       I[2] = I[1]
       I[1] = J
      C   if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
32     IF (I[2] .LE. I[3]) GOTO 33
       J = I[3]
       I[3] = I[2]
       I[2] = J
      C   if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
33     IF (I[3] .LE. I[4]) GOTO 34
       J = I[4]
       I[4] = I[3]
```

```
       I[3] = J
      C   if (a[4] > a[5]) { tmp = a[5] ; a[5] = a[4]; a[4] = tmp; }
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
34     IF (I[4] .LE. I[5]) GOTO 35
       J = I[5]
       I[5] = I[4]
       I[4] = J
      C
      C   if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
35     IF (I[0] .LE. I[1]) GOTO 36
       J = I[1]
       I[1] = I[0]
       I[0] = J
      C   if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
36     IF (I[1] .LE. I[2]) GOTO 37
       J = I[2]
       I[2] = I[1]
       I[1] = J
      C   if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
37     IF (I[2] .LE. I[3]) GOTO 38
       J = I[3]
       I[3] = I[2]
       I[2] = J
      C   if (a[3] > a[4]) { tmp = a[4] ; a[4] = a[3]; a[3] = tmp; }
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
38     IF (I[3] .LE. I[4]) GOTO 39
       J = I[4]
       I[4] = I[3]
       I[3] = J
      C
      C   if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
39     IF (I[0] .LE. I[1]) GOTO 40
       J = I[1]
       I[1] = I[0]
       I[0] = J
      C   if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
40     IF (I[1] .LE. I[2]) GOTO 41
       J = I[2]
       I[2] = I[1]
       I[1] = J
      C   if (a[2] > a[3]) { tmp = a[3] ; a[3] = a[2]; a[2] = tmp; }
```

```
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
41     IF (I[2] .LE. I[3]) GOTO 42
       J = I[3]
       I[3] = I[2]
       I[2] = J
      C
      C   if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
42     IF (I[0] .LE. I[1]) GOTO 43
       J = I[1]
       I[1] = I[0]
       I[0] = J
      C   if (a[1] > a[2]) { tmp = a[2] ; a[2] = a[1]; a[1] = tmp; }
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
43     IF (I[1] .LE. I[2]) GOTO 44
       J = I[2]
       I[2] = I[1]
       I[1] = J
      C
      C   if (a[0] > a[1]) { tmp = a[1] ; a[1] = a[0]; a[0] = tmp; }
      C   /* { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; */
44     IF (I[0] .LE. I[1]) GOTO 45
       J = I[1]
       I[1] = I[0]
       I[0] = J

45     RETURN
```

9 H: Change of Platform

Platform issues are going to be a hard repository problem. This section tests a range of platforms.

9.1 Case H1: Linux

⟨*CaseH1*⟩≡

9.2 Case H2: Windows

⟨*CaseH2*⟩≡

9.3 Case H3: IA86 Mac

⟨*CaseH3*⟩≡

10 I: Change of Control Structures

At this point we move away from the linear bubble sort and start to stress our ability to handle different control structures. We probably want to develop many more incremental steps of refinement of these cases.

10.1 Case I1: Loops

This is the original loop version of the bubble sort. It does no input or output and it works on a fixed length array.

⟨*CaseI1*⟩≡

```
#include <stdio.h>
int main()
{ int a[10] = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9};
  int swap = 1; /* true */
  int i = 0;
  int n = 9;
  int temp = 0;
  while (swap == 1) {
    swap = 0;
    i = 0;
    while (i < n) {
      if (a[i] < a[i+1]) {
        temp = a[i+1];
        a[i+1] = a[i];
        a[i] = temp;
        swap = 1;
      }
      i = i+1;
    }
  }
}
```

10.2 Case I2: Dynamic Data Length

Here we allow the size of the array to be chosen at run time so that we cannot compute the upper bound of the loop in a static manner.

⟨*CaseI2*⟩≡

10.3 Case I3: Recursion

Here we don't use looping at all. Rather we depend on the stack state of memory for control flow.

⟨*CaseI3*⟩≡

10.4 Case I4: Fork

Here we deal with the case of multiple threads of control, forking and joining in order to compute the result.

⟨*CaseI4*⟩≡

11 J: Change of Algorithm

There are many sort routines. Here we propose a few.

11.1 Case J1: Insertion Sort

This is a "simple" sort. We should be able to recreate similar test suites for linear and looping version of this sort.

⟨*CaseJ1*⟩≡

11.2 Case J2: Quicksort

Here we have an implicit use of recursion in a sort.

⟨*CaseJ2*⟩≡

⟨*⟩≡

```
all: code doc

code: loop.pamphlet
        @echo extracting sources
        @${TANGLE} -R"code" loop.pamphlet >loop.lisp
        @clisp -norc -x '(progn (load "loop.lisp") \
         (saveinitmem "isa" :executable t :norc t :quiet t \
         :init-function (lambda () (process-args) (ext:exit))))' >/dev.null
        @${TANGLE} -R"nasm" loop.pamphlet >test.asm
        @${ASM} test.asm
        @${DISASM} test.o > test.disasm

doc: loop.pamphlet axiom.sty
        @echo extracting loop.tex document
        @${WEAVE} -t8 -delay loop.pamphlet >loop.tex
        @echo latexing loop.dvi document
        @if [ -z "${NOISE}" ] ; then \
          ${LATEX} loop.tex ; \
         else \
          ${LATEX} loop.tex 2>/dev/null 1>/dev/null ; \
         fi
        @${MAKEINDEX} loop.idx 2>/dev/null 1>/dev/null
        @${LATEX} loop.tex 2>/dev/null 1>/dev/null
        @rm -f loop.log loop.toc loop.aux
        @rm -f loop.idx loop.ilg loop.ind

axiom.sty:
        @${TANGLE} -R"axiom.sty" loop.pamphlet >axiom.sty

remake:
        @echo Makefile rebuilt
        @mv Makefile.loop Makefile.bak
        @${TANGLE} -t8 loop.pamphlet >Makefile.loop

clean:
        @echo cleaning
```

References

[1] IA-32 Intel Architecture Software Developer's Manual Volume 2A: Instruction Set Reference, A-M Order Number 253666-016 June 2005,

[2] IA-32 Intel Architecture Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z Order Number 253667-016 June 2005