ENCM 339 Fall 2015 Exam Solutions

Steve Norman
Department of Electrical & Computer Engineering
University of Calgary

December 2016

Introduction

These solutions were prepared in December 2016, without access to notes used to mark exams in December 2015. It's possible that there are some mistakes here!

SECTION 1

1. (c) (The value of `*b` is `&a[0]` and the value of `b[1]` is `&a[2]`.)
2. (b) (The type of `cs` is `const char *`, so `cs` can't be used to change the value of a `char`.)
3. (d)
4. (b)
5. (c)
6. (c)
7. (b) or (d) (The correct answer was accidentally listed twice.)
8. (a) (`doubled(y)` gets preprocessed into `(y-1 * 2)`, and `*` has higher precedence than the `-` operator.)
9. (d) (The constructor call is equivalent to `Point p1(100, -99);`.)
10. (c)

SECTION 2

Question 1. 8 bytes.

Question 2. 1. (`fwrite` returns the number of items written; this information could be found in the *Reference Material*.)

Question 3. `'0'`, `'.'`, `'6'`, `'6'`, `'6'`, `'6'`, `'6'`, `'6'`.

Question 4.

```
char * copystr (const char* source) {
  char *dest;

  dest = malloc(strlen(source) + 1);

  strcpy(dest, source);
  return dest;
}
```

Question 5.

```
8
9
10
```

SECTION 3

Part 1.

```
bool up_then_down(const int* arr, int n) {
  if (n == 1)
    return true;

  int i_of_max = 0;
  for (int i = 0; i < n; i++)
    if (arr[i] > arr[i_of_max])
      i_of_max = i;

  for (int i = 1; i <= i_of_max; i++)
    if (arr[i] <= arr[i - 1])
      return false;

  for (int i = i_of_max + 1; i < n; i++)
    if (arr[i] >= arr[i - 1])
      return false;

  return true;
}
```

Part 2.

```
bool all_diff(const char *left, const char *right)
{
  for (int i = 0; left[i] != '\0'; i++)
    for (int j = 0; right[j] != '\0'; j++)
      if (left[i] == right[j])
        return false;

  return true;
}
```

SECTION 4

Question 1.

```
IntVector::IntVector(const IntVector& src)
  : storeM(0), end_validM(0), end_storeM(0)
{
  if (src.size() == 0)
    return;
  storeM = new int[src.size()];
  end_storeM = end_validM = storeM + src.size();
  for (size_t i = 0; i < src.size(); i++)
    storeM[i] = src.storeM[i];
}
```

Question 2.

```
IntVector::~IntVector() {
  delete [ ] storeM;
}
```

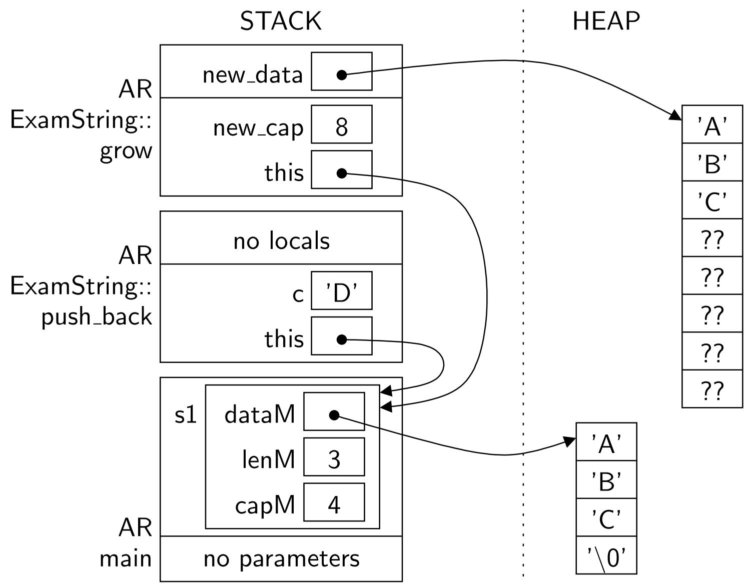
Question 3.

```
void IntVector::remove_all(int val) {
  size_t new_index = 0;
  for (size_t old_index = 0; old_index < size(); old_index++)
    if (storeM[old_index] != val) {
      storeM[new_index] = storeM[old_index];
      new_index++;
    }
  end_validM = storeM + new_index;
}
```

SECTION 5

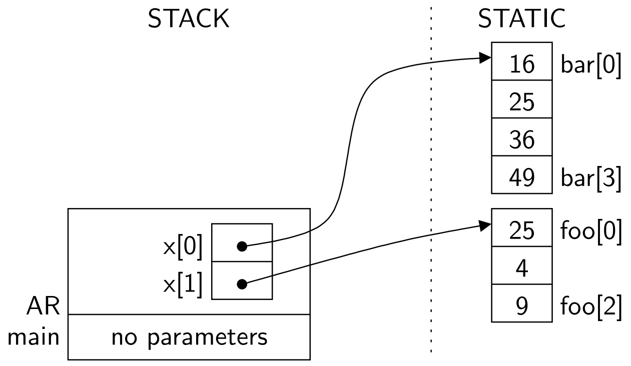
[This section was about linked lists, which is not a topic in ENCM 339 in Fall 2016.]

SECTION 6



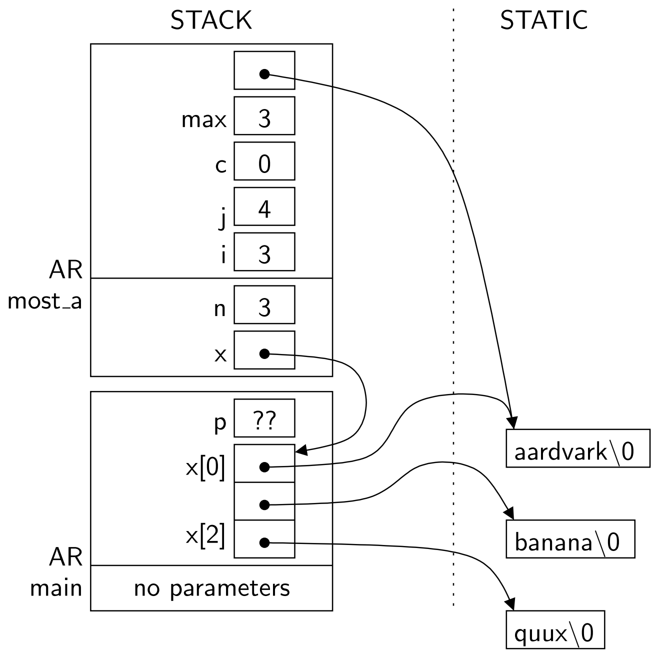
SECTION 7

Part a.



Part b.

(The style used here for drawing string constants is acceptable if pointers are pointing at the starts of strings, but not if a pointer points one or more `char`s away from the start of a string.)



SECTION 8

Part a.

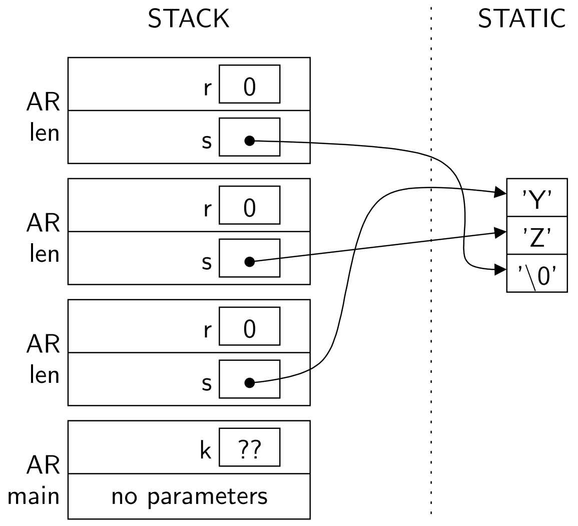
```
double max_brightness(const Image& im)
{
  double max_sum = 0.0;
  for (size_t r = 0; r < im.nrow(); r++) {
    for (size_t c = 0; c < im.ncol(); c++) {
      Pixel p = im.get_pixel(r, c);
      double sum = p.r + p.g + p.b;
      if (sum > max_sum)
        max_sum = sum;
    }
  }
  return max_sum / 765.0;
}
```

Part b.

```
Image mirror(const Image& im)
{
  Image result = Image(im.nrow(), im.ncol());
  for (size_t r = 0; r < im.nrow(); r++) {
    size_t im_c = im.ncol() - 1;
    for (size_t c = 0; c < im.ncol(); c++) {
      result.set_pixel(r, c, im.get_pixel(r, im_c));
      im_c--;
    }
  }
  return result;
}
```

SECTION 9

Part a.



Part b.

```
int first_match(const int *a, int lo, int hi, int key)
{
  int result;
  if (lo == hi - 1) {   // base case, only 1 element
    result = (a[lo] == key) ? lo : -1;
    return result;
  }

  int mid = (lo + hi) / 2;
  result = first_match(a, lo, mid, key);
  if (result == -1)
    result = first_match(a, mid, hi, key);

  return result;
}
```

SECTION 10

Binary file operations with the standard C++ library were not a topic in ENCM 339 in Fall 2016. But binary file operations with the standard C library were, so I thought it would be helpful to convert this into a C programming problem, then solve it ...

```
#include <stdio.h>
#include <stdlib.h> // for exit
#include <stdint.h> // for uint16_t

int main(int argc, char **argv) {
  if (argc != 2) {
    fprintf(stderr, "Error: need exactly one command-line argument.\n");
    exit(1);
  }
  FILE *fp = fopen(argv[1], "rb");
  if (fp == NULL) {
    fprintf(stderr, "Error: could not open %s for input.\n", argv[1]);
    exit(1);
  }

  char first4[4];
  uint16_t n_row, n_col;
  size_t nread = fread((void *) first4, 1, 4, fp);

  // Code above this comment is the C equivalent to the C++ code
  // given on the quetion paper.

  // Code below this comment is the solution to the C programming
  // version of this problem.

  if (nread != 4) {
    fprintf(stderr, "Error: could not read first 4 bytes of file.\n");
    exit(1);
  }
  if (first4[0] != 'i' || first4[1] != 'm'
      || first4[2] != 'g' || first4[3] != 'X') {
    fprintf(stderr, "Error: First 4 bytes of file are incorrect.\n");
    exit(1);
  }

  nread = fread((void *) &n_row, 1, sizeof(uint16_t), fp);
  if (nread == 2)
    nread = fread((void *) &n_col, 1, sizeof(uint16_t), fp);
  if (nread != 2) {
    fprintf(stderr, "Error: failed to read row and/or column counts.\n");
    exit(1);
  }
  printf("File seems to be OK, with %d rows and %d columns.\n",
         (int) n_row, (int) n_col);

  return 0;
}
```