16.216: ECE Application Programming

Fall 2014

Exam 2

November 5, 2014

Name: ____ **ID #:** __________________________

For this exam, you may use only one 8.5" x 11" double-sided page of notes. All electronic devices (e.g., calculators, cellular phones, PDAs) are prohibited. If you have a cellular phone, please turn it off prior to the start of the exam to avoid distracting other students.

The exam contains 3 questions for a total of 100 points. Please answer the questions in the spaces provided. If you need additional space, use the back of the page on which the question is written and clearly indicate that you have done so.

Please read each question carefully before you answer. In particular, note that:

- Question 3 has three parts, but you are only required to complete two of the three parts.
    - You may complete all three parts for up to 10 points of extra credit. If you do so, **please clearly indicate which part is the extra one—I will assume it is part (c) if you mark none of them.**
- For each part of Question 3, you must complete a short function. I have provided comments to describe what your function should do and written some of the code for you.
    - Note that each function contains both lines that are partially written (for example, a `printf()` call in which you are responsible for filling in the format string and expressions) and blank spaces in which you must write additional code. **You must write all code required to make each function work as described—do not assume you can simply fill in the blank lines and get full credit.**
    - Each function is accompanied by one or more test cases. Each test case is an example of how the function should behave in one specific case—**it does not cover all possible results of using that function.**
- You can solve each part of Question 3 using only the variables that have been declared, but you may declare and use other variables if you want.

You will have 1 hour to complete this exam.

Q1: Multiple choice	/ 20
Q2: Arrays	/ 40
Q3: Functions	/ 40
TOTAL SCORE	/ 100
EXTRA CREDIT	/ 10

1. (20 points, 5 points per part) ***<u>Multiple choice</u>***

For each of the multiple choice questions below, clearly indicate your response by circling or underlining the one choice you think best answers the question.

a. Given strings `s1 = "Exam 1"`, `s2 = "Exam 2"`, and `s3 = "Exactly"`, which of the following calls to string comparison functions will return the value 0?

   A. `strcmp(s1, s2);`
   B. `strncmp(s1, s2, 5);`
   C. `strcmp(s1, s3);`
   D. `strncmp(s1, s3, 3);`

   i. Only A

   ii. Only D

   iii. A and C

   iv. B and D

   v. All of the above (A, B, C, and D)

b. Given the short code sequence below:

```
int i;
char str[20] = "x";
for (i = 0; i < 4; i++)
  strcat(str, "ox");
printf("Length of string = %d\n", strlen(str));
```

What will this program print?

   i. `Length of string = 1`

   ii. `Length of string = 3`

   iii. `Length of string = 5`

   iv. `Length of string = 9`

   v. `Length of string = 20`

1 (continued)

c. Given the code sequence below:

```
char s1[20];
char s2[20];
strcpy(s1, "Summer");
s1[1] = 'i';
strncpy(s2, s1, 4);
s2[3] = '\0';
printf("%s  %s\n", s1, s2);
```

What will this program print?

i. `Summer  Simmer`

ii. `Simmer  Simmer`

iii. `Summer  Simm`

iv. `Simmer  Simm`

v. `Simmer  Sim`

d. Which of the following statements accurately reflect your opinion(s)? Circle all that apply (but please don't waste too much time on this "question")!

i. "I think the most recent programming assignments are still pretty easy."

ii. "I think the programming assignments have gotten to be too difficult."

iii. "I think the programming assignments have gotten harder, but are still fair."

iv. "Is the semester over yet?"

2. (40 points) ***Arrays***
For each short program shown below, list the output exactly as it will appear on the screen. Be sure to clearly indicate spaces between characters when necessary.

You may use the available space to show your work as well as the output; just be sure to clearly mark where you show the output so that I can easily recognize your final answer.

a. (12 points)

```
void main() {
  int i;
  double list[6] = { 1.2, 3.4, 5.6, 7.8, 9.0 };

  // Each value is printed with precision of one
  for (i = 0; i < 3; i++) {
    printf("%.1lf %.1lf\n", list[i], list[5 - i]);
  }

  for (i = 5; i > 0; i--) {
    printf("%.1lf\n", list[i] - list[i - 1]);
  }
}
```

2 (continued)

b. (14 points)

```
void main() {
  int i, j;
  int arr[4][2];

  for (i = 0; i < 4; i++) {
    for (j = 0; j < 2; j++) {
      arr[i][j] = i + j;
      printf("%d ", arr[i][j]);
    }
    printf("\n");
  }

  for (i = 7; i >= 0; i--) {
    printf("%d ", arr[i % 4][i % 2]);
  }
}
```

2 (continued)

c. (14 points)

```
void f(int arr[], int n) {
  int i;
  for (i = 0; i < n / 2; i++) {
    arr[n - 1 - i] += arr[i];
  }
}

void main() {
  int i;
  int list[] = { 0, 2, 4, 6, 8, 9, 7, 5, 3, 1 };

  f(list, 10);
  for (i = 0; i < 10; i++)
    printf("%d ", list[i]);
  printf("\n");

  f(list, 6);
  for (i = 0; i < 10; i++)
    printf("%d ", list[i]);
  printf("\n");

  f(list, 2);
  for (i = 0; i < 10; i++)
    printf("%d ", list[i]);
  printf("\n");
}
```

3. (40 points, 20 per part) ***Functions***

For each part of this problem, you are given a short program to complete. **CHOOSE ANY TWO OF THE THREE PARTS** and fill in the spaces provided with appropriate code. **You may complete all three parts for up to 10 points of extra credit, but must clearly indicate which part is the extra one—I will assume it is part (c) if you mark none of them.**

Remember, you must write all code required to make each function work as described—**do not assume you can simply fill in the blank lines and get full credit.** Also, remember that each example provided is only applicable in one specific case—**it does not cover all possible results of using that function.**

a. `void splitDay(int doy, int *m, int *dom);`

Given an integer, `doy`, representing the day within a year of 365 days, determine the appropriate month and day and store them in the variables pointed to by `m` and `dom`, respectively. For example, calling `splitDay(10, &month, &day)` would set `month = 1` and `day = 10`, since the 10th day of the year is January 10; `splitDay(360, &month, &day)` would set `month = 12` and `day = 26`, as the 360th day of the year is December 26.

```
void splitDay(int doy, int *m, int *dom) {
   int daysPerMo[] = { 31, 28, 31, 30, 31, 30,     // Days in
                       31, 31, 30, 31, 30, 31 };  //  each month
   int sum;     // Sum of days after a certain number of months

   // Initialize variables




   // Find the month by adding the number of days in all previous
   //   months--when you've added too many days, the previous
   //   month is the correct one
   while (____________________________________________________) {




   }

   // Finalize month and day values--day of month is what's left
   //   in the partial month after you've added all prior months




}
```

3 (continued)
b. `int countFib(int lo, int hi);`

This function counts the number of values in the Fibonacci sequence that fall between the endpoints lo and hi, including the endpoints. The first two Fibonacci numbers are 0 and 1, and every subsequent value in the sequence is the sum of the two numbers that precede it: **0, 1, 1, 2, 3, 5, 8, 13, 21, 34,** and so on. Therefore, for example, `countFib(1,10)` would return 6 and `countFib(1, 35)` would return 9.

```
int countFib(int lo, int hi) {
  int num1, num2;   // Numbers in Fibonacci sequence
  int count;        // Counter

  // Initialize variables




  // Special case: if lo is 0 or negative, count one more value
  if (________________________________________________________)



  // Go through Fibonacci values below low endpoint before
  //   starting count
  while (____________________________________________________) {






  }

  // Now generate and count Fibonacci values between endpoints
  while (____________________________________________________) {






  }

  return count;
}
```

3 (continued)

c. `int findMode(int arr[], int n);`

This function finds and returns the mode—the value that occurs most often—in an integer array `arr[]` of length `n`. For example, if `arr = {1, 2, 3, 3, 4, 4, 4}`, the mode is 4; if `arr = {1, 6, 2, 1, 6}`, the mode is 6.

To find the mode, the function counts the number of occurrences of each value in `arr[]`, and then determines the maximum number of occurrences, which in turn determines the mode. You may assume `arr[]` contains only values between 1 and 10, so the `numCount[]` array declared in the function can be used to count the occurrences of all possible values.

```
int findMode(int arr[], int n) {
   int numCount[10]; // Counts number occurrences in arr
   int i;            // Loop index
   int mode;              // Mode

   // Initialize variables as needed




   // Count # of occurrences of each value in arr


   for (_________________; ___________________; ______________) {




   }

   // Determine mode--find which number occurred most often


   for (_________________; ___________________; ______________) {





   }
   return mode;
}
```