EECE.2160: ECE Application Programming

Fall 2018

Exam 2
November 5, 2018

Name: _____

Lecture time (circle 1): *8-8:50 (Sec. 201)* *12-12:50 (Sec. 203)* *1-1:50 (Sec. 202)*

For this exam, you may use only one 8.5" x 11" double-sided page of notes. All electronic devices (e.g., calculators, cell phones) are prohibited. Please turn off your cell phone ringer prior to the start of the exam to avoid distracting other students.

The exam contains 3 sections for a total of 100 points, plus a 10 point extra credit question. 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:

- Questions 1b and 2c require you to complete short functions. We have provided comments to describe what each function should do and written some of the code.
    - Note that each function contains both lines that are partially written (for example, a `printf()` call missing 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 simply fill in the blank lines.**
    - Each test case is an example of how the program should behave in one specific case—**it does not cover all possible results of running that program.**
    - You can solve each of these questions using only the variables that have been declared, but you may declare and use other variables if you want.
- Carefully read the multiple choice problems. Questions 2b and 3a may have more than one correct answer, while questions 3b and 3c each have exactly one correct answer.
- **You cannot earn any extra credit without partial solutions to all parts of Sections 1, 2, and 3.** In other words, don't try Question 4 until you've attempted every other question on the exam—"finishing the exam" doesn't have to include the extra credit problem!

You will have 50 minutes to complete this exam.

S1: Functions	/ 35
S2: Arrays	/ 41
S3: Strings	/ 24
TOTAL SCORE	/ 100
EXTRA CREDIT	/ 10

1. (35 points) ***Functions***

a. (15 points) Show the output of the short program below 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 I can easily recognize your final answer.

```
int f1(int *arg1) {
   (*arg1)--;
   return (*arg1) * 3;
}

int f2(int arg2) {
   ++arg2;
   return arg2 + 10;
}

int main() {
   int a, b, c, d;
   a = 5;
   b = f1(&a);
   c = f2(a);
   d = f1(&b);
   a = f2( f1(&c) );

   printf("%d %d %d %d\n", a, b, c, d);
   return 0;
}
```

1 (continued)

b. (20 points) Complete the function described below:

```
double approx(double x, int n);
```

This function should calculate the following series approximation for the value 1 / (1-x), which is valid if the absolute value of x is less than 1:

$$\frac{1}{1-x} \approx \sum_{k=0}^{n} x^k = \mathbf{1} + \boldsymbol{x} + \boldsymbol{x^2} + \boldsymbol{x^3} + \cdots + \boldsymbol{x^n}$$

The function takes two arguments—the values of *x* and *n*, as shown above—and should return the approximate value calculated. For example, if $x = 0.5$ and $n = 3$, the function should return:

$$1 + 0.5 + 0.5^2 + 0.5^3 = 1 + 0.5 + 0.25 + 0.125 = 1.875$$

```
double approx(double x, int n) {
   double total;     // Running total for approximation
   double x_i;       // x to the power of i
   int i;            // Loop index

   // Initialize variables as needed









   // Loop to calculate series approximation as described above

   _____ (_____________________________________________________) {









   }

   // Return result

   return ___________________;
}
```

2. (41 points) ***Arrays***

a. (15 points) Show the output of the short program below 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 I can easily recognize your final answer.

```
int main() {
   int i, k;
   int vals[8] = {21, 60, 1, 6, -5, 7};
   int inds[8] = {3, 1, -1, 6, 8, 0, 10, 5};

   for (i = 6; i >= 0; i -= 2) {
      vals[i] += inds[i];
      printf("%d %d\n", vals[i], vals[i+1]);
      printf("%d %d\n", inds[i], inds[i+1]);
   }

   for (i = 0; i < 8; i++) {
      k = inds[i];
      if (k >= 0 && k < 8)
         printf("%d\n", vals[k]);
      else
         printf("%d\n", k);
   }

   return 0;
}
```

2 (continued)

b. (6 points) You are given a two dimensional array declared as: `double arr[5][10];`

You also have two `int` variables, `i` and `j`, and two variables of type `double`, `sum` and `avg`.

Which of the following short code sequences correctly calculate the average of all values in this array and stores it in the variable `avg`? **This question has at least one correct answer, but may have more than one correct answer! Circle ALL choices that correctly answer the question.**

i.
```
sum = 0;
for (i = 0; i < 5; i++) {
   for (j = 0; j < 10; j++) {
      sum += arr[i][j];
   }
   avg += sum / 10;
}
```

ii.
```
sum = 0;
for (i = 4; i >= 0; i--) {
   for (j = 9; j >= 0; j--) {
      sum += arr[i][j];
   }
}
avg = sum / 50;
```

iii.
```
sum = 0;
for (i = 0; i < 10; i++) {
   for (j = 0; j < 5; j++) {
      sum = sum + arr[j][i];
   }
}
avg = sum / 50;
```

iv.
```
sum = 0;
for (i = 0; i < 5; i++) {
   for (j = 0; j < 10; j++) {
      sum = arr[i][j];
   }
}
avg = sum / 50;
```

2 (continued)

c. (20 points) Complete the function described below:

```
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 (may need loop)









   // Use loop to count # of occurrences of each value in arr

   _____ (_____________________________________________________) {









   }
   // Loop to determine mode--which number occurred most often

   _____ (_____________________________________________________) {









   }
   return mode;
}
```

3. (24 points, 6 points each) ***Strings***

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? **This question has at least one correct answer, but may have more than one correct answer! Circle ALL choices that correctly answer the question.**

i. `strcmp(s1, s2);`

ii. `strncmp(s1, s2, 5);`

iii. `strcmp(s1, s3);`

iv. `strncmp(s1, s3, 3);`

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? **Choose only one answer.**

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`

3 (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? **Choose only one answer.**

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?"

4. (10 points) ***EXTRA CREDIT***

REMEMBER, YOU CANNOT EARN EXTRA CREDIT WITHOUT WRITING AT LEAST PARTIAL SOLUTIONS FOR ALL OTHER PROBLEMS ON THE EXAM.

However, you can earn partial credit for a partial solution to this problem.

Write the function with the function prototype and description below:

```
void removeChar(char *str, char c);
```

This function should remove all occurrences of the character `c` from the string `str`. Remember that, since the string is passed by address, changes made to `str` inside the function are seen outside the function.

For example, if the string `s1 = "This is a message"` before each function call shown below, then:

- After calling `removeChar(s1, 'i')` → `s1 = "Ths s a message"`
- After calling `removeChar(s1, ' ')` → `s1 = "Thisisamessage"`
- After calling `removeChar(s1, 's')` → `s1 = "Thi i a meage"`

Hint: to access part of a string (a substring) starting after the first character, use the address of the desired starting character. For example, given the original `s1` above, `&s1[5]` can be used to access the substring `"is a message"`. Using substrings with built-in string functions can simplify your solution.

Use the space on the next page to write your solution to this function.

4 (continued) ***SPACE TO SOLVE EXTRA CREDIT PROBLEM***

```
void removeChar(char *str, char c)
{

}
```