EECE.2160: ECE Application Programming

Fall 2017

Exam 2
November 8, 2017

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. If you have a cell phone, please turn off your ringer prior to the start of the exam to avoid distracting other students.

The exam contains 3 questions 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 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 running that function.**
    - 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.
- To ensure you do not ignore the main part of the exam in favor of the extra credit question, **you cannot earn any extra credit without writing at least partial solutions for all parts of Questions 1, 2, and 3.** In other words, don't try the extra credit question until you've attempted to solve every other question on the exam.

You will have 50 minutes to complete this exam.

Q1: Functions	/ 35
Q2: Arrays	/ 45
Q3: For loops; strings	/ 20
TOTAL SCORE	/ 100
Q4: 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 f(int a1, int *a2) {
   int temp = *a2;
   *a2 = a1 * 2;
   a1 = temp;
   return a1 / 2;
}

int main() {
   int v1, v2, v3, v4;

   v1 = 20;
   v2 = 30;
   v3 = f(v1, &v2);
   printf("%d %d %d\n", v1, v2, v3);
   v4 = f(v2, &v3);
   printf("%d %d %d %d\n", v1, v2, v3, v4);

   return 0;
}
```

1 (continued)

b. (20 points) Complete the function described below: `void squareGrid(int nBox);`

This function prints a square grid, where the number of boxes in each row and column of the grid is determined by the input argument `nBox`. For each iteration of the "row" loop, the function actually prints two rows of output: a set of `'+'` and `'-'` characters for the top or bottom part of a set of boxes, and then a set of `'|'` and `' '` characters for the boxes' middle part.

For example, calling `squareGrid(3)` would generate the following output:

```
+-+-+-+
| | | |
+-+-+-+
| | | |
+-+-+-+
| | | |
+-+-+-+
```

```
void squareGrid(int nBox) {
   int i, j;    // Loop indexes

   // Outer loop to control number of rows

   for (_____________________________________________________) {

      // Loop to print top/bottom of boxes

      for (_____________________________________________________) {


           printf("+-");


           printf("+\n");
      }

      // For all rows except last one, print middle of boxes

      if (_____________________________________________________) {

         for (_____________________________________________________) {


              printf("| ");


              printf("|\n");
         }
      }
   }
}
```

2. (45 points) ***Arrays***

a. (12 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 arr[6] = {9, 3, 4};
   int i;

   for (i = 5; i >= 0; i--) {
      printf("%d ", arr[i]);
      if (arr[i] == 0)
         arr[i] = arr[5-i];
      else
         arr[i] = arr[i+1] - 1;
   }
   printf("\n");
   for (i = 0; i < 3; i++)
      printf("%d %d\n", arr[i], arr[i+3]);
   return 0;
}
```

2 (continued)

b. (13 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. ***NOTE: Both printf() calls that print values from list[][] use a precision of 1.***

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.

```
int main() {
   int i, j;
   double list[2][4] = { {1.2, 3.4, 5.6, 7.8},
                         {9.8, 7.6, 5.4, 3.2} };

   for (i = 1; i >= 0; i--) {
      for (j = 3; j >= 0; j--) {
         printf("%.1lf ", list[i][j]);         // Precision of 1
      }
      printf("\n");
   }

   for (i = 0; i < 8; i++)
      printf("%.1lf\n", list[i%2][i%4]);       // Precision of 1
   return 0;
}
```

2 (continued)

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

```
int mostWins(int p1[], int p2[], int n);
```

This function takes three arguments: arrays `p1` and `p2`, which represent the scores of games between two players, and an integer `n`, which represents the number of games. The function should determine if player 1 or player 2 won more games and return the winning player's number. If both players won the same number of games, the function returns 0. For example:

- `mostWins({1,1,1}, {3,-1,2}, 3)` returns 2 (P2 won 2 of 3 games)
- `mostWins({0,1,2,3}, {-1,-2,-3,-4}, 4)` returns 1 (P1 won 4 of 4 games)
- `mostWins({1,2,3}, {3,2,1}, 3)` returns 0 (Each player won 1 game and tied 1 game, so neither player won more games than the other)

```
int mostWins(int p1[], int p2[], int n) {
   int i;            // Loop index
   int p1W, p2W;     // Wins for each player

   // Initialize variables



   // Go through arrays and determine winner of each match,
   //  keeping track of the total number of wins for each player

   for (_____________________________________________________) {

      // Player 1 won


      // Player 2 won


   }

   // Return number of winning player or 0 if tied


      return 1;


      return 2;


      return 0;
}
```

3. (20 points, 5 points each) ***For loops; strings***

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. What is the output of the short code sequence below?

```
int i;
int val = 1;
for (i = 0; i < 5; i += val) {
   val += i;
   printf("%d %d ", i, val);
}
```

i. `0 1`

ii. `1 1 2 3`

iii. `0 1 2 3 4`

iv. `0 1 1 2 3 5`

v. `0 1 1 2 2 4 3 7 4 11`

b. Given two strings, `s1 = "Exam 2"` and `s2 = "Exasperated"`, which of the following function calls will return 0?

i. `strlen(s1);`

ii. `strcmp(s1, s2);`

iii. `strncmp(s1, s2, 3);`

iv. `strncat(s1, s2, 3);`

v. All of the function calls above return non-zero values

3 (continued)

c. What is the output of the short code sequence below?

```
char s1[50] = "Q";
char s2[50] = "3";
int i;

for (i = 0; i < 3; i++) {
   strcat(s1, s2);
   strncpy(s2, s1, i + 1);
   s2[i+1] = '\0';        // Ensure s2 is null terminated
                          // Does not affect final output
}
printf("%s %s\n", s1, s2);
```

i. `Q 3`

ii. `Q3 Q`

iii. `Q3Q Q3`

iv. `Q3QQ3 Q3Q`

v. `Q3QQ3Q3Q Q3QQ`

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:

```
unsigned int substrMatch(char *s1, char *s2, unsigned int len,
                         unsigned int *pos1, unsigned int *pos2);
```

A substring is a short string within a larger string that can be defined by its starting position and length. For example, given the string `char s[] = "Example"`, a substring of length 3 starting at position 2 within `s` is `"amp"`.

The `substrMatch()` function searches its two string arguments, `s1` and `s2`, to see if both strings contain a matching substring of length `len`.

- If a match exists, the function returns 1, and the starting positions of the matching substring within `s1` and `s2` are stored in the variables pointed to by `pos1` and `pos2`, respectively.
- If no match exists, the function returns 0, and the variables pointed to by `pos1` and `pos2` are unchanged.

Notes and hints:

- Given a string `s[]`, you can access a substring at position `i` using `&s[i]`. For example, given `char s[] = "Example"`, `printf(&s[2]);` would print `ample`
- If there are multiple matching substrings of the desired length in `s1` and `s2`, your function should "find" the substring that appears earliest in `s1`.
- The only string functions you are allowed to use in your solution are the ones we discussed in class: `strcpy()`/`strncpy()`, `strcmp()`/`strncmp()`, `strlen()`, and `strcat()`/`strncat()`.

Test cases:

Given `char q[] = "strictest"`, `char r[] = "test"`, and unsigned ints `p1`, `p2`:

- `substrMatch(q, r, 2, &p1, &p2)` returns 1, with `p1 = 0` and `p2 = 2`
  (a substring of length 2 (`"st"`) is found at position 0 in `q` and position 2 in `r`)
- `substrMatch(q, r, 3, &p1, &p2)` returns 1, with `p1 = 5` and `p2 = 0`
  (a substring of length 3 (`"tes"`) is found at position 5 in `q` and position 0 in `r`)
- `substrMatch(q, r, 5, &p1, &p2)` returns 0
  (no matching substring of length 5 exists in the 2 strings)

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

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

```
unsigned int substrMatch(char *s1, char *s2, unsigned int len,
                         unsigned int *pos1, unsigned int *pos2)
{









}
```