EECE.2160: ECE Application Programming

Spring 2018

Exam 3

May 10, 2018

Name: ____

Lecture time (circle 1): *8-8:50 (Sec. 201)* *12-12:50 (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 5 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 2b and 3 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 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.
- For this exam, unlike Exams 1 and 2, you may attempt the extra credit problem even if you have not at least partially completed all other problems on the exam.
- The extra pages accompanying the exam hold structure and function definitions, as well as hints for the extra credit question—info needed for questions 2a, 2b, and 6.

You will have 3 hours to complete this exam.

Q1: Strings	/ 13
Q2: Structures	/ 33
Q3: File I/O	/ 20
Q4: Bitwise operators	/ 14
Q5: Multiple choice	/ 20
TOTAL SCORE	/ 100
Q6: EXTRA CREDIT	/ 10

1. (13 points) ***Strings***

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() {
   char str1[20];
   char str2[30];
   int n;

   strcpy(str1, "EECE.2160");
   strncpy(str2, "Spring 2018 Section 201", 11);
   str2[11] = '\0';

   printf("%s %s\n", str1, str2);

   n = strlen(str1);
   printf("str2[%d] = %c\n", n, str2[n]);

   strncat(str2, str1, 4);
   strncat(str1, str2, 4);
   printf("%s\n%s\n", str1, str2);

   return 0;
}
```

2. (33 points) ***Structures***

a. (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.

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.

The `Res` and `ResPair` structure definitions, as well as the `setRes()` function definition, are in the extra document provided with the exam.

```
int main() {
   Res res1 = {10, 2, 5};
   ResPair rp;

   setRes(&rp.r1, res1.r, 2, res1.r / res1.i);
   setRes(&rp.r2, res1.v * rp.r1.r, res1.v, rp.r1.r);

   printf("res1: %.2lf %.2lf %.2lf\n", res1.v, res1.i, res1.r);
   printf("rp.r1: %.2lf %.2lf %.2lf\n",
          rp.r1.v, rp.r1.i, rp.r1.r);
   printf("rp.r2: %.2lf %.2lf %.2lf\n",
          rp.r2.v, rp.r2.i, rp.r2.r);

   return 0;
}
```

2 (continued)
b. (20 points) Complete the function below (<u>the `Box` structure is defined on the extra sheet</u>):

```
int maxVol(Box list[], int n);
```

This function takes as arguments an array of `Box` structures, `list`, as well as the number of structures in the list, `n`. The function returns the index of the structure representing the box with the greatest volume. For example, given:

```
  Box arr[4] = { {1, 3, 5}, {9, 9, 9}, {4, 3, 2}, {2, 2, 2} };
```

the function call `maxVol(arr, 4)` would return 1, as `arr[1]` has the greatest volume (729, which is greater than the volumes of `arr[0]` (volume 15), `arr[2]` (24), and `arr[3]` (8)).

```
int maxVol(Box list[], int n) {
   int i;            // Loop index
   int maxI;         // Index of largest prism
   double maxVol;    // Volume of largest prism

   // Initialize variables as needed






   // Go through list; update max variables if larger box found

   for (_______________________________________________________) {

      if (_____________________________________________________) {







      }
   }

   // Return index of box with greatest volume

   return ________________________________;
}
```

3. (20 points) ***File I/O***

Complete the function described below:

```
int *readBin(char *fname);
```

This function dynamically allocates and returns the address of an array that stores the contents of a binary input file with name `fname`.

The file is formatted so that the first integer value in the file is the number of remaining values; in other words, a file intended to store the six integers 0, 1, 2, 3, 4, and 5 would actually contain the values 6, 0, 1, 2, 3, 4, and 5.

If the file cannot be opened or the space for the array cannot be allocated, the function should return NULL.

```
int *readBin(char *fname) {
   FILE *fp;           // File pointer
   int *arr;           // Pointer to array
   int n;              // Array size

   // Open binary input file and check that it opens correctly

   fp = fopen(________________________, _______________________);

   if (______________________________) {    // fopen unsuccessful
      printf("Could not open file\n");
      return NULL;
   }

   // Read first value from file, which gives size of array, and
   // dynamically allocate array. Return NULL if allocation fails



   arr = (int *)malloc(________________________________________);

   if (______________________________) { // Allocate unsuccessful
      printf("Could not allocate array\n");
      return NULL;
   }

   // Read remainder of file into array and return array address



   return arr;
}
```

4. (14 points) ***Bitwise operators***

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.

```
void main() {
   unsigned int x = 0x0000019F;
   unsigned int v1, v2, v3, v4;

   v1 = x & 0x00000FF0;
   v2 = x | 0x00000660;
   v3 = x ^ 0x11111111;
   v4 = x & ~(0x0000000F << 4);
   printf("%x\n", x);
   printf("%X\n", v1);
   printf("%.5x\n", v2);
   printf("%#X\n", v3);
   printf("%#.8x\n", v4);
}
```

5. (20 points, 4 points each) ***Multiple choice***

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

a. You have the following variables:

```
double *d;      // Dynamically allocated double array
int n;          // Current size of array
```

If `d` points to a dynamically allocated array holding `n doubles`, which of the following statements will reallocate space for this array so that the size of the array is divided in half without changing the remaining elements? In other words, if the array currently holds 10 elements, it will be shrunk to hold only 5 elements; those 5 elements will remain unchanged from the original array values.

i. `d = d / 2;`

ii. `d = (double *)malloc(n / 2 * sizeof(int));`

iii. `d = (double *)calloc(n / 2, sizeof(int));`

iv. `d = (double *)realloc(d, n / 2 * sizeof(int));`

v. `d = int [n/2];`

5 (continued)

b. Which of the following code sequences show an example of a memory leak?

A.
```
int *x = (int *)malloc(10 * sizeof(int));
int *y = x;
free(y);
```

B.
```
int *p1 = (int *)malloc(10 * sizeof(int));
int *p2 = (int *)malloc(10 * sizeof(int));
p1 = p2;
```

C.
```
int *x = (int *)malloc(10 * sizeof(int));
free(x);
x = NULL;
```

D.
```
int *p1 = (int *)malloc(10 * sizeof(int));
int *p2 = (int *)malloc(10 * sizeof(int));
free(p1);
p1 = p2;
```

i. Only A

ii. Only B

iii. A and C

iv. B and D

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

5 (continued)

c. You are writing a program that should repeatedly read input characters until a space is entered, then read the rest of the line that follows (up to 49 total characters) into an array:

Which of the following code sequences correctly read this input?

i.
```
char c;
char inp[50];
while ((c = getchar()) != ' ')
     ;    // Do nothing—empty loop
fgets(inp, 50, stdin);
```

ii.
```
char c;
char inp[50];
while ((c = getchar()) != ' ')
     ;    // Do nothing—empty loop
ungetc(c, stdin);
fgets(inp, 50, stdin);
```

iii.
```
char c;
char inp[50];
putchar(c);
fputs(inp, stdout);
```

iv.
```
char c;
char inp[50];
printf("%c %s\n", c, inp);
```

5 (continued)

d. Which of the following code snippets would flip the 8 highest bits (bits 24-31) and 8 lowest bits (bits 0-7) of an unsigned integer, x, without changing the middle 16 bits?

i. `x = x | 0xFF0000FF;`

ii. `x = x & 0x00FFFF00;`

iii. `x = x ^ 0xFF0000FF;`

iv. `x = x ^ 0x00FFFF00;`

v. None of the above

e. Circle one (or more) of the choices below that you feel best "answers" this "question."

i. "Thanks for the free points."

ii. "This is the best final exam I've taken today."

iii. "At least we're not here at 8:00 in the morning."

iv. None of the above.

6. (10 points) ***EXTRA CREDIT***

Everyone may attempt this problem, **even if you have not at least partially solved the rest of the exam.** Remember, you can earn partial credit for a partial solution to this problem.

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. *(Hint: The final output is easily readable, but you must show at least some work to get credit.)*

A definition for the `XCstruct` structure, as well as more hints, are on the extra handout.

```
int main() {
   XCstruct list[] = {  {4, 0x7AFE,    {6} },
                        {1, 0xA,       {0} },
                        {5, 0x6FEAF,   {2, 4} },
                        {6, 0xFFFFEF,  {3, 5, -3, -3, 2} } };
   char buf[25];
   int i, j, k;
   unsigned m, n, p;

   k = 0;
   for (i = 0; i < 4; i++) {
      m = 0x0000000F << (4 * (list[i].nch - 1));
      n = 4 * (list[i].nch - 1);
      p = 0;

      for (j = 0; j < list[i].nch; j++) {
         buf[k] = (list[i].val & m) >> n;

         if (buf[k] > 9 && buf[k] < 15)
            buf[k] += 87;
         else if (buf[k] == 15)
            buf[k] += 97 + list[i].add[p++];
         else
            buf[k] += 97;

         k++;
         n -= 4;
         m = m >> 4;
      }
      buf[k++] = ' ';
   }
   buf[0] -= 32;
   buf[k-1]++;
   buf[k] = '\0';
   printf("%s\n", buf);
   return 0;
}
```

6 (continued) ***<u>ADDITIONAL SPACE TO SOLVE EXTRA CREDIT PROBLEM</u>***