16.216: ECE Application Programming

Fall 2011

Exam 3

December 20, 2011

Name: ________________________________ ID #: _________________ Section: _________

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 the following about Question 3:

- The question has three parts, but you are only required to complete two of the three parts.
- For each part of that problem, you must complete a function. I have written the function prototype, as well as all of the input/output statements—you should not need to write any additional `printf()` or `scanf()` calls. You must fill in the rest of the function.
- You can solve each problem using only the variables that have been declared, but you may declare other variables if you want.
- **IMPORTANT:** For each part, you may write your solution either on the page where the problem is written, or on the following page (which gives you more space to work). For example, Question 3a may be solved on either page 10 or 11. **Do not fill in both pages.**

You will have 3 hours to complete this exam.

Q1: Multiple choice	/ 24
Q2: Arrays and pointers	/ 40
Q3: Arrays (1-D, 2-D, strings) and functions	/ 36
TOTAL SCORE	/ 100

1. (24 points, 4 points per part) ***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 a file, `output.txt`, that contains the following data:

```
1 2 3
abc
```

Which of the following code sequences could have produced this file?

i.
```
FILE *fp;
fp = fopen("output.txt", "r");
fscanf(fp, "%d %d %d", x, y, z);
fscanf(fp, "%s", str);
fclose(fp);
```

ii.
```
FILE *fp;
fp = fopen("output.txt", "w");
printf("%d %d %d\n", x, y, z);
printf("%s\n", str);
fclose(fp);
```

iii.
```
FILE *fp;
fp = fopen("output.txt", "w");
fprintf(fp, "%d %d %d\n", x, y, z);
fprintf(fp, "%s\n", str);
fclose(fp);
```

iv.
```
fprintf(stdout, "%d %d %d\n", x, y, z);
fprintf(stdout, "%s\n", str);
```

1 (cont.)

b. You have a program that contains an array declared as:

```
double arr[40];
```

Which of the following code snippets would correctly store the contents of this array in a binary file?

i.
```
FILE *fp = fopen("output.txt", "w");
fread(arr, sizeof(double), 40, fp);
```

ii.
```
FILE *fp = fopen("output.txt", "w");
fprintf(fp, "%lf", arr);
```

iii.
```
FILE *fp = fopen("output.txt", "w");
fwrite(arr, sizeof(double), 40, fp);
```

iv.
```
FILE *fp = fopen("output.txt", "w");
printf("The contents of this array in a binary file\n");
```

1 (cont.)

The following question uses the structure defined below:

```
typedef struct {
  char name[50];
  double progs[8];
  double exams[3];
} CourseData;
```

c. Which of the following choices is <u>not</u> a valid access to a field within a variable of type `CourseData`?

i.
```
CourseData cd;
cd.progs[7] = 95;
```

ii.
```
CourseData list[85];
list[0]->exams[0] = 83;
```

iii.
```
CourseData s1;
scanf("%s", s1.name);
```

iv.
```
typedef struct {
  CourseData students[49];
} SectionData;

SectionData sd;
sd.students[0].exams[1] = 75;
```

1 (cont.)
The following question uses the structure defined below:

```
typedef struct {
   char name[50];
   double progs[8];
   double exams[3];
} CourseData;
```

d. Which of the following choices show valid uses of a pointer to access a field within a variable of type `CourseData`?

A.
```
CourseData cd;
CourseData *p = &cd;
p->exams[2] = 85;
```

B.
```
double exAvg(CourseData *c) {
   return (c->exams[0] + c->exams[1] + c->exams[2]) / 3;
}
```

C.
```
CourseData a[10];
CourseData *p = a;
printf(p->name);
```

D.
```
CourseData b[15];
CourseData *p1 = &b[0];
CourseData *p2 = &b[14];
while (p1 != p2) {
   scanf("%lf", &p1->progs[0]);
   p1++;
}
```

i. A and C
ii. A and B
iii. B and C
iv. B and D
v. A, B, C, and D

1 (cont.)

e. Why are structures typically passed by address, not by value, when they are used as function arguments?

i. Structures have to be passed by address—the C language doesn't allow you to copy the entire contents of a structure at once.

ii. Functions that deal with structures always need to be able to change the structure inside the function, which can't be done if the structure is passed by value.

iii. Passing a structure by address uses less memory than passing it by value, since structures typically contain multiple fields and therefore use more space than a scalar variable.

iv. Passing a structure by value would cause a rift in the space-time continuum.

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

i. 12

ii. "Thanks for the free points."

iii. "I don't REALLY have to answer the last two questions, do I?"

iv. "Of all the tests I've taken that started at 6:30 PM today, this one is my favorite."

v. "I don't think it's fair that Dr. Geiger can sit there and play Angry Birds while we're suffering through this test."
(I'm not actually playing Angry Birds … as far as you know.)

2. (40 points) ***Arrays and pointers***
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.

I encourage you to 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. (13 points)

```c
int main() {
     int x[10] = {0};
     int *ptr = x;
     int i;

     ptr[2] = 16;
     ptr[1] = 216;
     ptr[6] = 2011;

     ptr = ptr + 5;

     ptr[0] = 12;
     ptr[1] = 20;
     ptr[-2] = 3;

     ptr = ptr - 2;
     ptr[4] = ptr[5] = ptr[6] = 201;

     for (i = 0; i < 10; i++)
          printf("%d ", x[i]);
     printf("\n%d\n", *ptr);
     return 0;
}
```

2 (cont.)

b. (13 points)

```
int main() {
     char str[] = "Applications";
     char *ptr1, *ptr2;
     char temp;

     ptr1 = str;
     ptr2 = &str[11];

     while (*ptr1 != '\0') {
          temp = *ptr1;
          *ptr1 = *ptr2;
          *ptr2 = temp;
          ptr1 += 4;
          ptr2 -= 3;
     }

     printf("%s\n", str);
     return 0;
}
```

2 (cont.)

c. (14 points)

```
int main() {
     int orig[] = {75, 82, 13, 90, 51, 58, -1};
     int scaled[6];
     int *po = orig;
     int *ps = scaled;
     int i;

     while (*po != -1) {
          *ps = *po + 6;
          po++;
          ps++;
     }

     for (i = 0; i < 6; i++) {
          po--;
          ps--;
          printf("%d %d\n", *po, *ps);
     }
     return 0;
}
```

3. (36 points, 18 per part) ***Arrays (1-D, 2-D, and strings) and functions***
For each part of this problem, you are given a function to complete. Note that some of the code is provided for you. **CHOOSE ANY TWO OF THE THREE PARTS** and fill in the spaces with appropriate code. **If you complete all three, I will grade only the first two.**

a. `void countOccurrences(int occ[], int n);`

- Given: an array, `occ[]`, and the array length, `n`.
  - Note: You must correctly initialize `occ[]` before using it.
- Repeatedly prompt for and read integers from standard input and count how many times each value is entered, using `occ[]`—`occ[0]` is the number of times 0 is entered, `occ[1]` is the number of times 1 is entered, and so on.
- Exit the function when the input value is not a valid array index for `occ[]`.
- Example: assume n = 4, and user enters: `0 0 0 1 1 2 2 2 2 2 3 -1`
  - `occ[0] = 3, occ[1] = 2, occ[2] = 5, occ[3] = 1`
  - Prompts (`"Enter integer value: "`) not shown in example to save space

```
void countOccurrences(int occ[], int n) {
     int inval;      // Input value
     int i;          // Loop index









          // Prompt for and read integer
          printf("Enter integer value: ");
          scanf("%d", &inval);




               return;    // Input value is invalid--exit function









}
```

ALTERNATE SPACE TO ANSWER QUESTION 3A
<u>IF YOU SOLVED THE PROBLEM ON THE PREVIOUS PAGE, DO NOT FILL IN THIS PAGE</u>

```
void countOccurrences(int occ[], int n) {
     int inval;      // Input value
     int i;          // Loop index









          // Prompt for and read integer
          printf("Enter integer value: ");
          scanf("%d", &inval);







               return;    // Input value is invalid--exit function









}
```

3 (cont.)

b. `void matrixMul(int nR, double d[][3], double m1[][2],`
`                 double m2[][3]);`

- Given: three 2-D arrays, `d`, `m1`, and `m2`, and one array dimension, `nR`.
    - o Size of `d`: `nR x 3`
    - o Size of `m1`: `nR x 2`
    - o Size of `m2`: `2 x 3`
- Perform the matrix multiplication `m1 * m2` and store the results in `d`.
- Example: In the example below, `nR == 2`:

$$\overset{\texttt{m1}}{\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & 4 \\ \hline\end{array}} * \overset{\texttt{m2}}{\begin{array}{|c|c|c|}\hline 5 & 6 & 7 \\ \hline 8 & 9 & 10 \\ \hline\end{array}} = \overset{\texttt{d}}{\begin{array}{|c|c|c|}\hline 1*5 + 2*8 = 21 & 1*6 + 2*9 = 24 & 1*7 + 2*10 = 27 \\ \hline 3*5 + 4*8 = 47 & 3*6 + 4*9 = 54 & 3*7 + 4*10 = 61 \\ \hline\end{array}}$$

- Hint: The problem may be slightly easier to solve if you take the example above and replace each value with an array name and index—for example, the value 1 is actually `m1[0][0]`—to get a better idea of how to calculate each element of `d`.

```
void matrixMul(int nR, double d[][3], double m1[][2], double m2[][3]){
     int i;          // Loop variable for rows of m1 and d
     int j;          // Loop variable for columns of m2 and d









}
```

ALTERNATE SPACE TO ANSWER QUESTION 3B

IF YOU SOLVED THE PROBLEM ON THE PREVIOUS PAGE, DO NOT FILL IN THIS PAGE

```
void matrixMul(int nR, double d[][3], double m1[][2], double m2[][3]);
     int i;          // Loop variable for rows of m1 and d
     int j;          // Loop variable for columns of m2 and d





}
```

3 (cont.)

c. `int removeSub(char str[], int start, int end);`

- Given: a string `str`, and positions within that string, `start` and `end`.
  - You can assume `start` and `end` do not need to be checked for errors.
  - Neither position references the null character at the end of `str`.
- Remove characters from `str` between positions `start` and `end`, including those two positions, and shift the remaining characters so you now have a shorter string.
- Return the number of characters in the new string.
- Examples: Assume you have an array `char s[] = "Test string";`
  - `removeSub(s, 0, 0)` → `returns 10; s = "est string"`
  - `removeSub(s, 0, 4)` → `returns 6; s = "string"`
  - `removeSub(s, 4, 7)` → `returns 7; s = "Testing"`

```
int removeSub(char str[], int start, int end) {
   int pos1, pos2;         // Positions within string str[]









}
```

ALTERNATE SPACE TO ANSWER QUESTION 3C

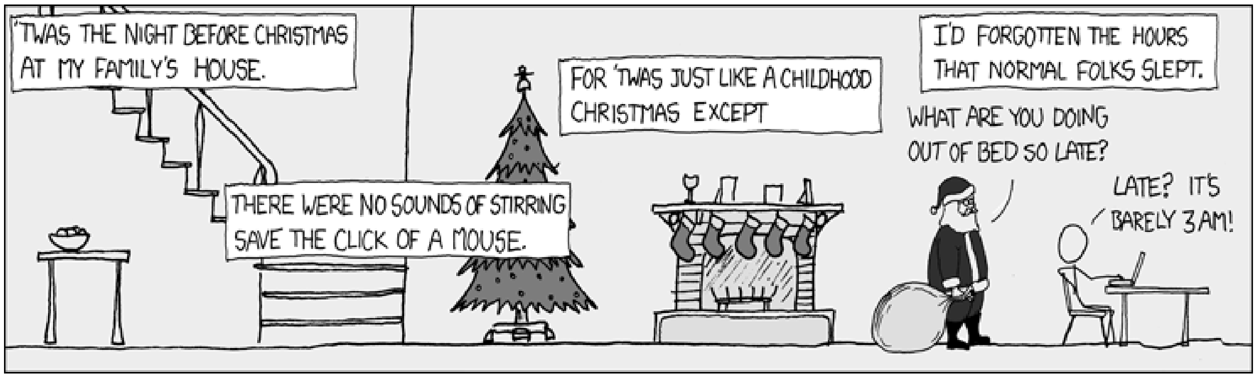
<u>IF YOU SOLVED THE PROBLEM ON THE PREVIOUS PAGE, DO NOT FILL IN THIS PAGE</u>

```
int removeSub(char str[], int start, int end) {
   int pos1, pos2;          // Positions within string str[]






}
```


'TWAS THE NIGHT BEFORE CHRISTMAS AT MY FAMILY'S HOUSE.
THERE WERE NO SOUNDS OF STIRRING SAVE THE CLICK OF A MOUSE.
FOR 'TWAS JUST LIKE A CHILDHOOD CHRISTMAS EXCEPT
I'D FORGOTTEN THE HOURS THAT NORMAL FOLKS SLEPT.
WHAT ARE YOU DOING OUT OF BED SO LATE?
LATE? IT'S BARELY 3 AM!

http://xkcd.com/361/

I hope you enjoy the holidays, the winter break, and, if you're like most engineering students I know, the chance to catch up on some much-needed sleep.

--Dr. Geiger