16.216: ECE Application Programming

Summer 2012

Exam 3

August 16, 2012

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 that problem, you must complete a function. I have written part of the program for you and provided comments to describe what each missing piece of code should do.
- You can solve each problem using only the variables that have been declared, but you may declare and use other variables if you want.
- Note that you may, if you need more space, complete each of these problems on the back of another page—just clearly note where your solution is.

You will have three hours to complete this exam.

Q1: Multiple choice	/ 20
Q2: Strings; pointers	/ 40
Q3: Arrays and functions	/ 40
TOTAL SCORE	/ 100
EXTRA CREDIT	/ 10

1. (20 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:

```
abc 0 3 4
```

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

i.
```
FILE *fp;
int x = 0, y = 3, z = 4;
char str[10] = "abc";
fp = fopen("output.txt", "r");
scanf("%s %d %d %d", str, &x, &y, &z);
fclose(fp);
```

ii.
```
FILE *fp;
int x = 0, y = 3, z = 4;
char str[10] = "abc";
fp = fopen("output.txt", "w");
fprintf(fp, "%s", str);
fprintf(fp, "%d %d %d\n", x, y, z);
fclose(fp);
```

iii.
```
FILE *fp;
int x = 0, y = 3, z = 4;
char str[10] = "abc";
fp = fopen("output.txt", "w");
fwrite(str, x, y, z);
fclose(fp);
```

iv.
```
int x = 0, y = 3, z = 4;
char str[10] = "abc";
printf("%s %d %d %d\n", str, x, y, z);
```

1 (cont.)

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

```
double arr[20];
```

Which of the following code snippets would correctly read the contents of this array from a file?

i.
```
FILE *fp = fopen("input.txt", "r");
fscanf(fp, "%lf", arr);
```

ii.
```
FILE *fp = fopen("input.txt", "r");
fread(fp, sizeof(double), 20, arr);
```

iii.
```
FILE *fp = fopen("input.txt", "r");
fread(arr, sizeof(double), 20, fp);
```

iv.
```
FILE *fp = fopen("input.txt", "r");
fwrite(arr, sizeof(double), 20, fp);
```

1 (cont.)

c. You are writing a program that should repeatedly read input characters until a space is entered, then read the space and 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);
```

1 (cont.)

d. The following question uses the structure defined below:

```
typedef struct {
   int number;
   char name[40];
   char rating[7];
   int length;
   char time[4][7];
} Movie;
```

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

i.
```
Movie m;
scanf("%s", m->name);
```

ii.
```
typedef struct {
   Movie mList[10];
} TheaterData;

TheaterData td;
td.mList[0].length = 120;
```

iii.
```
Movie m;
m.length = 123;
```

iv.
```
Movie TDKR;
Movie *P = &TDKR;
strcpy(P->name, "The Dark Knight Rises");
```

1 (cont.)

e. Choose your favorite statement(s) from the list below. Circle all that apply (but don't waste too much time!):

   i. "I'm so glad I spent the last six weeks taking this course—that's how I've always wanted to spend July and August!"

      (You may be more likely to agree with this statement if you imagine saying it with just a hint of sarcasm.)

   ii. "The exam ends here, right? We don't REALLY need to do the last two questions … "

   iii. "You're going to drop our lowest program score or two … or five … right?"

   iv. "I'd actually like to take this opportunity to leave some constructive feedback in the space below" *(Note: don't do this unless you have time left at the end of the exam.)*

2. (40 points) ***Strings; 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.

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, n;
     char *p;
     char str[] = "thisgridhidesone!";
     p = &str[5];
     n = 1;
     for (i = 0; i < 6; i++) {
          printf("%c", *p);
          p = p + n;
          n = n * -2;
     }
}
```

2 (cont.)

b. (14 points)

```
void main() {
     char s1[20] = "";
     char s2[20] = "";
     strcat(s1, "ab");
     strcat(s2, "ac");
     printf("%s %s\n", s1, s2);
     strcat(s1, s2);
     strcat(s2, s1);
     printf("%s %s\n", s1, s2);
     strncat(s1, s2, 3);
     strncat(s2, s1, 3);
     printf("%s %s\n", s1, s2);
}
```

2 (cont.)

c. (14 points)

```c
void main() {
     char s1[10] = "baaaad";
     char s2[10] = "b";
     int i = 1;

     do {
          printf("%s %s\n", s1, s2);
          strncat(s2, "a", 1);
          i++;
     } while (strncmp(s1, s2, i) == 0);
}
```

3. (40 points, 20 per part) ***Arrays and functions***
For each part of this problem, you are given a function 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.**

a. `int checkIfSorted(int a[], int n);`

Given an array of integer values, `a[]`, and the size of the array, `n`, check if the array is sorted from smallest to largest value. If so, return 1; if not, return 0. For example, if:

- `a = {1, 2, 3, 4, 5}` → `checkIfSorted(a, 5)` returns 1
- `a = {1, 2, 4, 3, 5, 7, 8}` → `checkIfSorted(a, 7)` returns 0

Solution:

```
int checkIfSorted(int a[], int n) {
     int i;               // Loop variable

     // GO THROUGH ARRAY—AS SOON AS YOU FIND A PAIR OF
     //   INCORRECTLY SORTED VALUES, RETURN 0









          // TEST CONDITION THAT SHOWS ARRAY IS NOT SORTED
          //   AND RETURN 0 IF TRUE






               return 0;









     return 1;            // IF FUNCTION REACHES THIS POINT,
                          //   ARRAY MUST BE SORTED
}
```

3 (cont.)

b. `int longestMatch(char *s1, char *s2);`

Given two strings, s1 and s2, return the length of the longest matching character sequence between the two, starting with the first character of each. For example:

- `longestMatch("string", "other")` returns 0
- `longestMatch("string", "stuff")` returns 2
- `longestMatch("string", "strings")` returns 6

```
int longestMatch(char *s1, char *s2) {
     int nC;    // Current string length being tested

     // INITIALIZE VARIABLES IF NEEDED





     // KEEP TESTING STRINGS UNTIL MATCH IS NOT FOUND
     //   OR YOU'VE REACHED END OF STRING S1







          // STRINGS DON'T MATCH—RETURN LENGTH OF LONGEST
          //   MATCHING STRING




               return nC;

          // UPDATE nC




     }

     // IF YOU REACH THIS POINT, ALL CHARACTERS UP TO LENGTH
     //   OF S1 MATCH—RETURN nC
     return nC;
}
```

3 (cont.)

c. `int countZeroColumns(int arr[][10], int n);`

Given a 2-D integer array, `arr[][10]`, and the number of rows in the array, `n`, return the number of columns in this array in which most of the values are 0. For example, if the array has three rows, you count the number of columns that contain two or more zeroes.

```
int countZeroColumns(int arr[][10], int n) {
     int totalCols = 0;       // Total # of columns with
                              // mostly zero values
     int zeroesPerCol;        // # of zeroes in a given column
     int i, j;

     // GO THROUGH ALL COLUMNS





          // FOR EACH COLUMN, GO THROUGH ALL ROWS AND COUNT
          //    NUMBER OF ZEROES









          // AFTER CHECKING ALL ROWS IN A GIVEN COLUMN, CHECK
          //    TO SEE IF THE MAJORITY OF VALUES IN THAT COLUMN
          //    ARE ZERO—IF SO, UPDATE COUNT OF COLUMNS WITH
          //    MOSTLY ZERO VALUES








     return totalCols;        // Return count of columns with
                              //   mostly zeroes
}
```