16.216: ECE Application Programming

Spring 2012

Exam 3

May 15, 2012

Name: ________________________________ **ID #:** _________________ **Section:** ___201___

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: Functions and pointers	/ 40
Q3: Loops and switch statements	/ 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, `input.txt`, that contains the following data:

```
70.0 73.7 69.6
abc
```

Which of the following code sequences can be used to read input from this file?

i.
```
FILE *fp;
double x, y, z;
char str[10];
fp = fopen("input.txt", "r");
scanf("%lf %lf %lf %s", &x, &y, &z, str);
fclose(fp);
```

ii.
```
FILE *fp;
double x, y, z;
char str[10];
fp = fopen("input.txt", "w");
fprintf(fp, "%lf %lf %lf\n", x, y, z);
fprintf(fp, "%s\n", str);
fclose(fp);
```

iii.
```
FILE *fp;
double x, y, z;
char str[10];
fp = fopen("input.txt", "r");
fscanf(fp, "%lf %lf %lf %s", &x, &y, &z, str);
fclose(fp);
```

iv.
```
double x, y, z;
char str[10];
fscanf(stdin, "%lf %lf %lf %s", &x, &y, &z, str);
```

1 (cont.)

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

```
char list[50];
```

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, "%c", list);
```

ii.
```
FILE *fp = fopen("input.txt", "r");
fwrite(list, sizeof(char), 50, fp);
```

iii.
```
FILE *fp = fopen("input.txt", "r");
fread(list, sizeof(char), 50, fp);
```

iv. All of the above

1 (cont.)

c. You are writing a program that should accept data from the standard input, in the following format:
   - A single character, followed by a newline
   - An entire line of data that may contain spaces, as well as up to 50 characters

Which of the following code sequences can correctly read this input?

i.
```
char c;
char inp[50];
c = getchar();
getchar();                  // Remove newline
fgets(inp, 50, stdin);
```

ii.
```
char c;
char inp[50];
c = getchar();
ungetc(c, stdin);
fgets(inp, 50, stdin);
```

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

iv.
```
char c;
char inp[50];
c = fgetc(stdin);
fgetc(stdin);               // Remove newline
fgets(inp, 51, stdin);
```

1 (cont.)
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;
```

d. 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.rating);
```

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

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

iii.
```
Movie m;
m.name[0] = 'A';
```

iv.
```
Movie list[100];
strcpy(list[5].name, "Dude, Where's My Car?");
```

1 (cont.)

e. Which of the following statements most accurately describes your feelings at this point in the exam? Circle all that apply.

   i. “It’s nice to finally have a joke of a multiple choice question where the answers don’t look like code, so I know I’m not supposed to take the question seriously.”

   ii. “8:00 exams are terrible—it’s a good thing this exam doesn’t cover the whole semester, because I might have fallen asleep halfway through it.”

   iii. “Are we done yet?”

   iv. “I’m disappointed—none of these answers are actually funny.”

   v. All of the above

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. (14 points)

```
void main() {
     char str[] = "Show sample string\n";
     int v[] = {6, 10, -11, -2, 7, 4, 4};
     int i;
     char *p = str;

     for (i = 0; i < 7; i++) {
          p = p + v[i];
          printf("%c", *p);
     }
}
```

2 (cont.)

b. (12 points)

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

     strcpy(str1, "16.216");
     strncpy(str2, "Spring 2012 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);
}
```

c. (14 points)

```
void main() {
     char s1[] = "AbCdEfG";
     char s2[] = "AbCDEfg";
     int i = 1;

     while (i <= strlen(s1)) {
          if (strncmp(s1, s2, i) != 0) {
               printf("%d: N\n", i);
          }
          else
               printf("%d: Y\n", i);
          i++;
     }
}
```

3. (40 points, 20 per part) ***<u>Arrays and functions</u>***

For each part of this problem, you are given a function to complete. **<u>CHOOSE ANY TWO OF THE THREE PARTS</u>** and fill in the spaces provided with appropriate code. **<u>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.</u>**

a. `int checkPalindrome(char str[]);`

This function takes a single string, `str[]`, as input and checks if the input string is a palindrome—a word or phrase that is read the same in either direction. Examples include "dad", "noon", and "madam". The function should follow these specifications:

- Return 1 if the input string is a palindrome; return 0 otherwise.
- Assume any letters in `str[]` are lowercase letters, and that `str[]` contains no spaces.

Examples:

- `checkPalindrome("madam") == 1`
- `checkPalindrome("123321") == 1`
- `checkPalindrome("randomness") == 0`
- `checkPalindrome("modem") == 0`

```
int checkPalindrome(char str[]) {
     int i;               // Index variable
     int n;               // Total characters in str, other than '\0'
     int mid;             // Midpoint of string, which you may find
                          //   helpful in your solution

     // CALCULATE n, mid







     // LOOP TO COMPARE CHARACTERS FROM BEGINNING AND END OF STRINGS









                return 0;             // str[] is not a palindrome


     return 1;        // str[] is a palindrome
}
```

3 (cont.)

b. `double sumElements(int iVals[], int n1, double arr[], int n2);`

This function takes the following arguments:

- An integer array, `iVals[]`, that contains `n1` elements
- A double array, `arr[]`, that contains `n2` elements

The elements of `iVals[]` select elements from `arr[]` that are added together, assuming the index values are valid. The sum of these values is returned. For example, assume you have:

- `double L[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};`
- `int index1[] = {0, 1, 2, 3};`
- `int index2[] = {0, -1, 2, 11, 4};`

Using these arrays, you would get the following results:

- `sumElements(index1,4,L,10) = L[0] + L[1] + L[2] + L[3] =`
  `1 + 2 + 3 + 4 = 10`
- `sumElements(index2,5,L,10) = L[0] + L[2] + L[4] =`
  `1 + 3 + 5 = 9`
    - Note that, although `index2[]` contains 5 different potential index values, -1 and 11 are not valid indices—`L[-1]` and `L[11]` don't exist—and those values are therefore ignored.

```
double sumElements(int iVals[], int n1, double arr[], int n2) {
     double sum;                // Sum of appropriate values
     int i;                     // Index variable



     // INITIALIZE SUM, THEN LOOP THROUGH ALL ELEMENTS OF iVals[]






          // USE ONLY VALID VALUES FROM iVals[] TO SELECT ELEMENTS
          //   OF arr[] AND ADD THOSE VALUES TO THE SUM










     return sum;     // Return final sum
}
```

3 (cont.)

c. `void countRowNum(int vals[][10], int rowNumCount[], int nR);`

Given a two-dimensional array, `vals[][10]`, containing `nR` rows, go through each row and determine how many times the row number itself appears in that row. Store this count in `rowNumCount[]`. In other words, `rowNumCount[0]` indicates how many times the number 0 appears in row 0 of `vals[][]`, `rowNumCount[1]` indicates how many times the number 1 shows up in row 1, etc.)

For example, if you have the following arrays:

```
     int x[3][10] = { {0, 0, 0, 1, 2, 3, 4, 5, 6, 7},
                      {1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
                      {0, 2, 0, 2, 0, 2, 0, 2, 0, 2} };
     int y[3];
```

Calling `countRowNum(x,y,3)` will fill `y[]` with the values `{3,1,5}`, because, within `vals[][]`, row 0 contains three 0s, row 1 contains a single 1, and row 2 contains five 2s (as shown by the underlined values).

```
void countRowNum(int vals[][10], int rowNumCount[], int nR) {
     int i, j;        // Loop index variables



     // FOR EACH ROW IN vals[][], COUNT THE NUMBER OF TIMES THE
     //   CORRESPONDING ROW NUMBER IS SEEN AND STORE THE TOTAL
     //   OCCURRENCES IN THE APPROPRIATE ELEMENT OF rowNumCount[]









}
```