# 16.216: ECE Application Programming

Fall 2013

Exam 2 Solution

1. (20 points, 5 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 (continued)

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 binary file?

i.
```
FILE *fp = fopen("input.bin", "rb");
fscanf(fp, "%c", list);
```

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

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

iv. All of the above

1 (continued)

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?

***Note:*** *I've marked either (i) or (iv) as acceptable, as both have flaws:*

- *Choice (i) reads a maximum of 49 characters—remember, the second argument to the `fgets()` function is one higher than the maximum number of characters actually read.*
- *Choice (iv) reads the correct maximum of 50 characters, but the array `inp[50]` doesn't have enough space to hold 50 input characters and a terminating null character (`'\0'`).*

***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 (continued)

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?"

***All of the above are "correct."***

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

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() {
  double arr[6] = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6};
  int i;

  for (i = 5; i >= 0; i--)
    printf("%.1lf ", arr[i]);
  printf("\n");

  for (i = 0; i < 6; i += 2) {
    arr[i] = arr[i] + arr[i+1];
    printf("%.1lf ", arr[i]);
  }
  printf("\n");
}
```

**This loop prints the contents of arr[], starting with the last element**

**This loop modifies and prints only the even-numbered elements of arr[] (arr[0], arr[2], arr[4]). Each element is set as the sum of its original value and the next value in the array (e.g., arr[0] = arr[0] + arr[1])**

**OUTPUT:**

```
6.6 5.5 4.4 3.3 2.2 1.1
3.3 7.7 12.1
```

2 (continued)

b. (14 points)

```
void main() {
   int tab[2][3] = { {0,  5,  10},
                     {15, 20, 25} };
   int i, j;

   for (i = 0; i < 2; i++) {
      for (j = 0; j < 3; j++)
         printf("%d ", tab[i][j]);
      printf("\n");
   }

   for (i = 0; i < 6; i++)
      printf("%d ", tab[i/3][i/2]);
   printf("\n");
}
```

**This loop prints tab[][] one row at a time, in the exact order shown above**

**This loop prints six values from the array, with some values duplicated: tab[0][0], tab[0][0], tab[0][1], tab[1][1], tab[1][2], and tab[1][2]**

**OUTPUT:**

```
0 5 10
15 20 25
0 0 5 20 25 25
```

2 (continued)

c. (14 points)

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

     strcpy(str1, "16.216");        str1 = "16.216"

     strncpy(str2, "Fall 2013 Section 201", 11);
     str2[11] = '\0';               str2 = "Fall 2013 S" (second
                                    line adds terminating null
                                    that strncpy() doesn't add)

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

     n = strlen(str1);              n = 6
     printf("str2[%d] = %c\n", n, str2[n]);      str2[n] = '0'

     strncat(str2, str1, 4);        str2 = "Fall 2013 S16.2"
     strncat(str1, str2, 4);        str1 = "16.216Fall"
     printf("%s\n%s\n", str1, str2);
}
```

***Solution:*** *See above for how variables are changed throughout program*

***FINAL OUTPUT***

```
16.216 Fall 2013 S
str2[6] = 0
16.216Fall
Fall 2013 S16.2
```

3. (40 points, 20 per part) ***Functions***

For each part of this problem, you are given a short program 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 countDigits(int val);`

This function takes a single integer, `val`, as an argument and returns the number of digits in that integer. For example, `countDigits(5) = 1`; `countDigits(15932) = 5`.

The number of digits in the number can be found by repeatedly dividing `val` until the result is 0; the number of steps required to reach that point is the number of digits in `val`. For example, if `val = 16216`, the function goes through the sequence below to determine `val` has 5 digits:

16216 → 1621 → 162 → 16 → 1 → 0 (each arrow represents one step)

***Students were responsible for code written in bold, underlined, italicized font.***

```
int countDigits(int val) {
     int n;         // Number of digits in val

     // Initialize variables as needed
     n = 0;

     // Special case: return the value 1 when val is 0
     if (val == 0)
          return 1;

     // In all other cases
     // Repeatedly divide val and count the number of
     //   steps required to reach 0
     while (val != 0) {
          val = val / 10;
          n++;
     }

     // Return number of digits
     return n;
}
```

3 (continued)

b. `void split_time(int total_sec, int *hr, int *min, int *sec);`

This function reads in a value, `total_sec`, which represents the number of seconds since midnight, and splits that value into the appropriate number of hours (between 0-23, with 13 hours representing 1 PM, 14 representing 2 PM, etc.), minutes (0-59), and seconds (0-59).

The hours, minutes, and seconds should be stored in the values pointed to by `hr`, `min`, and `sec`, respectively. Remember that a minute contains 60 seconds, and an hour contains 60 minutes. The examples below show the results of calling this function, assuming variables `h`, `m`, and `s` are declared in the function that calls `split_time()`:

- `split_time(5, &h, &m, &s)      → h = 0, m = 0, s = 5`
- `split_time(60, &h, &m, &s)     → h = 0, m = 1, s = 0`
- `split_time(10000, &h, &m, &s) → h = 2, m = 46, s = 40`

***Students were responsible for code written in bold, underlined, italicized font.***

```
void split_time(int total_sec, int *hr, int *min, int *sec) {


     // Calculate the number of hours, then remove that much
     //   time from total_sec before calculating minutes/seconds
     *hr = total_sec / 3600;
     total_sec %= 3600;

     // Calculate the number of minutes and seconds
     *min = total_sec / 60;
     *sec = total_sec % 60;
}
```

3 (continued)

c. `void insertionSort(int arr[], int n);`

This function sorts `n` values in an integer array, `arr`, from lowest to highest, as follows:

1. Start with the second array element; copy it into a temporary variable (`temp`).
2. Compare `temp` to all values to the left (i.e., with a lower array index) until you find the appropriate spot for that value. At each step:
   a. If the value you're testing is greater than `temp`, move that value one spot to the right.
   b. Otherwise, place `temp` one spot to the right and go to step 3.
   c. If you test all array elements to the left of the current one and don't stop at step (b), `temp` should be the new first value in the array.
3. Copy the 3rd array element to `temp`; repeat steps 2a-2c. Continue with the 4th, 5th, etc.

The example below shows how the function would sort the array {5, 2, 3, 7}:

Step 1: `temp = 2` {5,2,3,7} → {5,**5**,3,7} → {**2**,5,3,7}

Step 2: `temp = 3` {2,5,3,7} → {2,5,**5**,7} → {2,**3**,5,7}

Step 3: `temp = 7` No sorting done, since 7 is greater than all values to its left

***Students were responsible for code written in bold, underlined, italicized font.***

```
void insertionSort(int arr[], int n) {
   int i, j;         // Array index variables
   int temp;         // Temporary value

   // Start with 2nd value and visit all remaining array elements

   for (i = 1; i < n; i++) {
      // Copy current array value to temp
      temp = arr[i];

      // Test values to the left of the current position until
      //   you find right spot for temp or have tested all values
      // Values greater than temp should be shifted right 1 spot
      j = i – 1;
      while((temp < arr[j]) && (j >= 0)) {
         arr[j+1] = arr[j];
         j = j-1;
      }

      // At this point, you've found right spot--copy temp there
      arr[j+1] = temp;
   }
}
```