16.216: ECE Application Programming

Summer 2012

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 are given the following short program:

```
int i = 10;
while (i > 0) {
   i--;
   if (i > 3)
      continue;

   else if ((i % 2) == 0)
      break;
}
```

How many iterations will this loop execute?

i. 10

ii. 8

iii. 7

iv. 3

v. An infinite number—the loop never ends

b. You are given the following function prototype:

```
double f(int a, int b);
```

In a program with integer variables `x` and `y`, and double-precision variable `d`, which of the following calls to function `f()` would not compile (i.e., which one is an incorrect call to `f()`)?

i. `f(x, y);`

ii. `x = f(y, y + 2);`

iii. ***`d = f(&y, &x);`***

iv. `d = f(5, 7);`

v. `d = f(-3, x);`

c. Given the following code snippet:

```
int x = 1;
int i = 0;
while (i <= 3) {
   x = x * 2;
   i++;
}
```

Which of the following choices can replace the `while` loop and produce the exact same value for `x`? Assume `x` is always initialized to 1.

i. `for (i = 1; i < 4; i++)`

ii. `for (i = 0; i < 3; i++)`

iii. `for (x = 0; x <= 3; x++)`

iv. ***`for (i = 3; i >= 0; i--)`***

v. None of the above

d. Which of the following "functions" would you find most useful? Circle all that apply *(and please don't waste too much time on this "question").*

i.
```
set_all_16_216_grades(100);
```

ii.
```
set_program_status(&program3, &program4, "graded");
```

iii.
```
print_answer_key_to_screen();
```

iv.
```
set_overall_GPA(4.0);
graduate_now();
```

All are "correct."

2. (40 points) ***Functions 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.

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() {
   int var1, var2, var3;
   int *pt1, *pt2, *pt3;

   pt1 = &var2;           pt1 points to var2
   pt2 = &var3;           pt2 points to var3
   pt3 = &var1;           pt3 points to var1

   var1 = 16;
   var3 = 216;
   *pt1 = *pt3 + 7;       *pt1 is var2; *pt3 is var1. Therefore:
                          var2 = var1 + 7 = 16 + 7 = 23

   pt1 = pt2;             pt1 now points to same value as pt2
                               → both point to var3

   *pt2 = *pt3 + *pt1;    *pt2 and *pt1 both refer to var3
                          *pt3 is var1   Therefore:
                          var3 = var1 + var3 = 16 + 216 = 232

   printf("%d %d %d\n", var1, var2, var3);
   printf("%d %d %d\n", *pt1, *pt2, *pt3);
}
```

Solution: *See above for how each assignment changes variables. Output is below:*

```
   16 23 232
   232 232 16
```

2 (cont.)

b. (12 points)

```
int mac(int v1, int v2, int v3) {
   return v1 * v2 + v3;
}

void main() {
   int r1, r2, r3;

   r1 = mac(2,2,7);       r1 = 2 * 2 + 7 = 4 + 7 = 11
   r2 = mac(-2,3,9);      r2 = -2 * 3 + 9 = -6 + 9 = 3
   r3 = mac(r1, r2, 7);   r3 = r1 * r2 + 7
                             = 11 * 3 + 7 = 33 + 7 = 40

   printf("%d %d %d\n", r1, r2, r3);
   printf("%d\n", mac(r3,r2,r1));   mac(r3,r2,r1) =
                                     40 * 3 + 11 = 120 + 11 = 131
   printf("%d\n", mac(r1,r2,r3));   mac(r1,r2,r3) =
                                     11 * 3 + 40 = 33 + 40 = 73

}
```

Solution:

```
   11 3 40
   131
   73
```

c. (14 points)

```
int swapIfGT(int *x, int *y) {
   int temp;

   if (*x > *y) {
      temp = *x;
      *x = *y;
      *y = temp;
      return 1;
   }

   return 0;
}
```

```
void printVars(int *a, int *b) {
   if (swapIfGT(a,b) == 1)
      printf("%d > %d\n", *a, *b);
   else
      printf("%d <= %d\n", *a, *b);
}
```

Function will swap values if first argument > second If that condition is true, output won't make sense

```
void main() {
   int v1, v2, v3, v4;
   v1 = 5;
   v2 = 7;
   v3 = 9;
   v4 = 1;

   printVars(&v1,&v2);
   printVars(&v3,&v4);
   printVars(&v2,&v3);
   printVars(&v1,&v4);
}
```

Since 5 < 7, no swap; prints "5 <= 7"

Since 9 > 1, values swapped (v3 = 1, v4 = 9) and output is "1 > 9"

Since 7 > 1, values swapped (v2 = 1, v3 = 7) and output is "1 > 7"

Since 5 < 9, no swap; prints "5 <= 9"

Solution:

```
5 <= 7
1 > 9
1 > 7
5 <= 9
```

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

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 space 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. Repeatedly prompt a user to enter two double-precision values and then read those numbers. Your program should end when the second number entered is less than the first—at that point, print the maximum value entered. A sample run is below; user input is underlined:

```
Enter two values: 1 3
Enter two values: -0.7 1.234
Enter two values: 55 55
Enter two values: 16.216 16.217
Enter two values: 2.3 -3.7
Max value: 16.217000
```

```
void main() {
   double val1, val2;    // Input values
   double max;           // Max value—you may assume, as Visual
                         //   Studio does, that this variable
                         //   initially holds the lowest
                         //   possible negative value if it's
                         //   not explicitly initialized

   // REPEATEDLY PROMPT USER TO ENTER TWO VALUES—MAY TEST END
   //   CONDITION HERE OR SPACE BELOW, DEPENDING ON LOOP TYPE
   do {
         printf("Enter two values: ");
         scanf("%lf %lf", &val1, &val2);

         // DETERMINE IF EITHER VALUE IS NEW MAX VALUE
         if (val1 > max)
               max = val1;
         if (val2 > max)
               max = val2;

   // MAY TEST LOOP END CONDITION HERE
   } while (val1 <= val2);
   printf("Max value: %lf\n", max);
}
```

b. Prompt for and read a series of characters and count the number of whitespace characters—spaces, tabs (`'\t'`) and newlines (`'\n'`)—in the input. Stop reading if the user enters the same non-whitespace character twice in a row. Print the total number of whitespace characters. The sample below contains 1 tab, 3 spaces, and 2 newlines (input is underlined):

```
ab    3 6 ?                    (Note: tab is between 'b' and '3')
h Q
zz
Total whitespace characters: 6
```

```
void main() {
      char inChar;          // Input value
      char lastChar;        // Input value from previous iteration
      char spaceCnt;        // # whitespace characters

      // INITIALIZE VARIABLES AS NEEDED
      inChar = ' ';    // DONE TO ENSURE LOOP DOESN'T EXIT EARLY
      spaceCnt = 0;

      // REPEATEDLY (1) SAVE RESULT OF PREVIOUS ITERATION, AND
      // (2) READ NEW CHARACTER
      while (1) {
            lastChar = inChar;
            scanf("%c", &inChar);

            // WHITESPACE CHARACTER FOUND—INCREMENT COUNT
            if ((inChar == ' ') || (inChar == '\t') ||
                (inChar == '\n'))
                  spaceCnt++;

            // INPUT CHARACTER ISN'T WHITESPACE AND MATCHES VALUE
            //   FROM PREVIOUS ITERATION--EXIT
            else if (lastChar == inChar)
                  break;
      }
      printf("Total whitespace characters: %d\n", spaceCnt);
}
```

c. Complete the program below so that it reads a positive, non-zero integer, n, then prints all values between 1 and n, including n. The output should be formatted as follows:
   - The first line of output contains a single value.
   - For all other lines, use the last value on the previous line to determine the number of values to print on the following line.

See the examples below for further clarification; user input is underlined.

```
Enter n: 8
1
2
3 4
5 6 7 8
```

```
Enter n: 14
1
2
3 4
5 6 7 8
9 10 11 12 13 14
```

```
void main() {
      int n;                  // Input value
      int valsPerLine;        // Values to be printed on the current line
      int numPrinted;         // Number of values printed already on the
                              // current line
      int i;                  // Loop index

      // INITIALIZE VARIABLES AS NEEDED
      valsPerLine = 1;
      numPrinted = 0;

      // Prompt for and read input values
      printf("Enter n: ");
      scanf("%d", &n);

      // LOOP TO PRINT ALL VALUES BETWEEN 1 AND n, INCLUDING n
      for (i = 1; i <= n; i++) {

            printf("%d ", i);

            // CHECK HOW MANY VALUES HAVE BEEN PRINTED
            //   ON CURRENT LINE. IF LIMIT HAS BEEN
            //   REACHED, MOVE TO NEXT LINE, AND
            //   UPDATE valsPerLine AND numPrinted
            numPrinted++;
            if (numPrinted == valsPerLine) {
                  printf("\n");
                  valsPerLine = i;
                  numPrinted = 0;
            }
      }
}
```