16.216: ECE Application Programming

Practice Problems: Loops
Solution

1. What does the following program print?

a.
```
int main() {
   int x = 0;
   while (x < 10) {
      printf("x = %d",  ++x);
      x++;
   }
   return 0;
}
```

Solution: *Note that, for obvious reasons, it would have been a good idea to put a '\n' at the end of the printf() format string.*

```
x = 1x = 3x = 5x = 7x = 9
```

1 (cont.) What does the following program print?

b.
```
int main() {
    int i, j;

    for (i = 0; i < 3; i++) {
       printf("i is %d\n", i);
       for(j = 0; j < 5; j++)
          printf("i is %d, j is %d\n", i, j);
       printf("end of i = %d loop\n", i);
    }
    return 0;
}
```

Solution:

```
i is 0
i is 0, j is 0
i is 0, j is 1
i is 0, j is 2
i is 0, j is 3
i is 0, j is 4
end of i = 0 loop
i is 1
i is 1, j is 0
i is 1, j is 1
i is 1, j is 2
i is 1, j is 3
i is 1, j is 4
end of i = 1 loop
i is 2
i is 2, j is 0
i is 2, j is 1
i is 2, j is 2
i is 2, j is 3
i is 2, j is 4
end of i = 2 loop
```

1 (cont.) What does the following program print?

c.
```
int main() {
    int x;
    int i = 0;

    for (x = 0; x <= 3; x++){
      printf("Start: x = %d, i = %d\n", x, i);
      x = x * 2;
      i++;
      printf("End: x = %d, i = %d\n", x, i);
    }
    printf("Final: x = %d, i = %d\n", x, i);
}
```

Solution:

```
Start: x = 0, i = 0
End: x = 0, i = 1
Start: x = 1, i = 1
End: x = 2, i = 2
Start: x = 3, i = 2
End: x = 6, i = 3
Final: x = 7, i = 3
```

1 (cont.) What does each of the following programs print?

d.
```
int main() {
     int x = 4;
     int n = 0;

     while (x > 5) {
        if (x == 10)
           x = 0;
        else
           x += 2;
        printf("x = %d\n", x);
        n++;
     }
     printf("n = %d\n", n);
     return 0;
  }
```

Solution:
```
   n = 0
```

e.
```
int main() {
     int x = 4;
     int n = 0;

     do {
        if (x == 10)
           x = 0;
        else
           x += 2;
        printf("x = %d\n", x);
        n++;
     } while (x > 5);
     printf("n = %d\n", n);
     return 0;
  }
```

Solution:
```
   x = 6
   x = 8
   x = 10
   x = 0
   n = 4
```

1 (cont.) What does the following program print?

```
f. int main() {
      int num = 625;

      while (num >= 1) {
         printf("num = %d\n", num);
         num /= 5;
      }
      return 0;
   }
```

Solution:

```
   num = 625
   num = 125
   num = 25
   num = 5
   num = 1
```

2. Write a program to do each of the following tasks:
(NOTE: You do not have to do any error checking in these programs unless the problem explicitly specifies that you do so.)

a. Print all multiples of 2 between 10 and 100, including the endpoints (i.e., print both 10 and 100).

Solution:

```
int main() {
     int i;                          // Loop variable

     for (i = 10; i <= 100; i+= 2) {
          printf("%d\n", i);         // Print current value of i
     }
     return 0;
}
```

b. Repeatedly prompt a user to enter two double-precision values, then read those numbers. Your program should end when the second number entered is less than the first—at that point, print `"Program complete"`. 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
Program complete
```

Solution:

```
int main() {
     double val1, val2;        // Input values

     // Repeatedly prompt user to enter two values
     do {
          printf("Enter two values: ");
          scanf("%lf %lf", &val1, &val2);

     } while (val1 <= val2);  // If val2 < val1, condition is
                              //   false and loop is done

     printf("Program complete\n");
     return 0;
}
```

2 (cont.)

c. Prompt for and read in a series of characters, stopping when the user enters the character `'q'`. Print the following outputs:
   - If the character is `'A'` or `'a'`, print `"Absolute value\n"`
   - If the character is `'C'` or `'c'`, print `"Cosine\n"`
   - If the character is `'S'` or `'s'`, print `"Sine\n"`
   - If the character is `'T'` or `'t'`, print `"Tangent\n"`
   - For all other characters, print `"Invalid input\n"`

Solution:

```
int main() {
     double inChar;                // Input values

     // Repeatedly prompt user to enter character
     do {
          printf("Enter single character: ");
          scanf("%c", &inChar);

          switch (inChar) {
          case 'A':
          case 'a':
               printf("Absolute value\n");
               break;
          case 'C':
          case 'c':
               printf("Cosine\n");
               break;
          case 'S':
          case 's':
               printf("Sine\n");
               break;
          case 'T':
          case 't':
               printf("Tangent\n");
               break;
          default:
               printf("Invalid input\n");
          }
     } while (inChar != 'q');

     return 0;
}
```

2 (cont)

d. Prompt for and read in a series of integers, and keep track of the largest and smallest values entered. Stop reading when the user enters a value outside the range 16 <= *n* <= 216; this final value should not be considered as the largest or smallest. After the user enters a value outside the range, print the largest and smallest values entered. A sample run is below:

```
Enter integer between 16 and 216: 17
Enter integer between 16 and 216: 216
Enter integer between 16 and 216: 53
Enter integer between 16 and 216: 1
Largest value: 216
Smallest value: 17
```

Solution:

```
int main() {
     int inval;          // Input value
     int max = 16;       // Max value
     int min = 216;      // Min value
                         // Initializing min/max to opposite
                         //   ends of range ensures that
                         //   actual input values should
                         //   overwrite these values

     // Repeatedly prompt user to enter integer
     do {
          printf("Enter integer between 16 and 216: ");
          scanf("%d", &inVal);

          // Input out of range--exit
          if ((inVal < 16) || (inVal > 216))
               break;

          // Set max and min as necessary
          if (inVal < min)
               min = inVal;
          if (inVal > max)
               max = inVal;

     } while (1);

     printf("Largest value: %d\n", max);
     printf("Smallest value: %d\n", min);
     return 0;
}
```

2 (cont.)

e. Prompt for and read in a series of characters and count the number of whitespace characters—spaces, tabs (‘\t’) and newlines (‘\n’)—in the list. Stop reading when the user enters the same non-space character twice in a row. Print the total number of whitespace characters. A sample run is below; it contains 1 tab, 3 spaces, and 2 newlines:

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

Solution:

```
int main() {
     char inChar = ' ';  // Input value
                         // Initialize to space to ensure loop
                         //    exit condition isn’t met in first
                         //    iteration
     char lastChar;      // Input value from previous iteration
     char spaceCnt = 0;  // # whitespace characters

     do {
          lastChar = inChar;
          scanf("%c", &inChar);

          // Whitespace character--increment count
          if ((inChar == ' ') || (inChar == '\t') ||
              (inChar == '\n'))
               spaceCnt++;

          // Input character isn’t whitespace, and it matches
          //    character from previous iteration—exit loop
          else if (inChar == lastChar)
               break;

     } while (1);

     printf("Total whitespace characters: %d\n", spaceCnt);
     return 0;
}
```

3. Write a program with a series of statements to produce each of the following patterns. Use only the following `printf()` statements. Each `printf()` may only appear once in each segment of the program. A sample template of the program appears on the next page.

```
printf("*");    printf(" ");    printf("\n");
```

I've posted my solution to this problem as a separate file: stars.c

a.
```
****
****
****
****
****
```

b.
```
*
***
*****
*******
```

c.
```
*****
****
***
**
*
```

d.
```
*
**
***
****
*****
******
*******
```

e.
```
*********
*********
*********
*********
*********
*********
*********
```

f.
```
*****
*******
*********
***********
*************
```

g.
```
*******
******
*****
****
***
****
*****
******
*******
```

h.
```
*************************
*****************
*********
****
*
```

i.
```
*************************
**********************
****************
**********
*
**********
****************
**********************
*************************
```

j.
```
*
***
*****
*******
*********
***********
*********
*******
*****
***
*
```

```
#include <stdio.h>
int main() {
   // declare variables as needed
   printf("----------------------------- Pattern 1\n");
   // code to produce pattern 1
   // printf(" "), printf("*"), and printf("\n")
   // may only appear once in this section
   printf("----------------------------- Pattern 2\n");
   // code to produce pattern 2
   // printf(" "), printf("*"), and printf("\n")
   // may only appear once in this section
   printf("----------------------------- Pattern 3\n");
   // code to produce pattern 3
   // printf(" "), printf("*"), and printf("\n")
   // may only appear once in this section
   printf("----------------------------- Pattern 4\n");
   // code to produce pattern 4
   // printf(" "), printf("*"), and printf("\n")
   // may only appear once in this section
   printf("----------------------------- Pattern 5\n");
   // code to produce pattern 5
   // printf(" "), printf("*"), and printf("\n")
   // may only appear once in this section
   // code and header for patterns 6, 7, 8
   printf("----------------------------- Pattern 9\n");
   // code to produce pattern 9
   // printf(" "), printf("*"), and printf("\n")
   // may only appear once in this section
   printf("----------------------------- Pattern 10\n");
   // code to produce pattern 10
   // printf(" "), printf("*"), and printf("\n")
   // may only appear once in this section

   return 0;
}
```