

COP 3223: C Programming Spring 2009

Control Structures - Revisited

Instructor : Dr. Mark Llewellyn
 markl@cs.ucf.edu
 HEC 236, 407-823-2790
<http://www.cs.ucf.edu/courses/cop3223/spr2009/section1>

School of Electrical Engineering and Computer Science
University of Central Florida



An Easy To Make Error With A For Statement

- What is the output from the following program, assuming the user enters the value 15?

```
bad for loop.c
1 //for loop with an infinite loop error
2 //January 29, 2009   Written by: Mark Llewellyn
3
4 #include <stdio.h>
5
6 int main()
7 {
8     int i; //a loop control variable
9     int upperLimit; //maximum number to be printed
10
11     printf("How many integers do you want to print?\n");
12     scanf("%d", &upperLimit);
13
14     for (i = 1; i <= upperLimit; i++); {
15         printf("The number is: %d\n", i);
16     } //end for stmt
17
18     printf("\n\n");
19     system("PAUSE");
20     return 0;
21 } //end main function
```



An Easy To Make Error With A For Statement

```
bad for loop.c
1 //for loop with an infinite loop error
2 //January 29, 2009   Written by: Mark J. Llewellyn
3
4 #include <stdio.h>
5
6 int main()
7 {
8     int i; //a loop control variable
9     int upperLimit; //maximum number to be printed
10
11     printf("How many integers do you want to print?\n");
12     scanf("%d", &upperLimit);
13
14     for (i = 1; i <= upperLimit; i++); {
15         printf("The number is: %d\n", i);
16     } //end for stmt
17
18     printf("\n\n");
19     system("PAUSE");
20     return 0;
21 } //end main function
```

```
C:\Courses\COP 3223 - C Programming\Spring 2009\COP 322...
How many integers do you want to print?
15
The number is: 16
Press any key to continue . . . . _
```

The problem is this semicolon which effectively ends the `for` statement. Thus, the body of the `for` loop contains no statements. The `for` loop executes properly and the value of `i` after the loop terminates is 16.



An Easy Error To Make With Conditional Expressions

- There is one type of error that many beginning C programmers make, including some experienced C programmers, when dealing with conditional expressions.
- This error is confusing the equality operator (==) with the assignment operator (=). Often this mistake is made as simply a typing error, but has potentially disastrous effects on your program, because it will ordinarily not generate a syntax error.
- To illustrate this problem, consider the case where we have a set of employees and their paycodes, where if the employee's paycode is set to 5, we will print a message that they have been awarded a bonus.
- Consider the two cases shown on the next page:



An Easy Error To Make With Conditional Expressions

```
if (payCode == 4) {  
    printf("You'll get a bonus!\n");  
}
```

 (a)

```
if (payCode = 4) {  
    printf("You'll get a bonus!\n");  
}
```

 (b)

- If we intended to type the version shown in (a), but mistakenly typed the version in (b), what will happen?
- Version (b) will evaluate the assignment expression , which simply assigns the constant value 4 to the variable `payCode`. Since C interprets any nonzero value as “true”, the condition of this if statement is always true and the person (indeed every person) will be awarded the bonus!



An Easy Error To Make With Conditional Expressions

GOOD PROGRAMMING PRACTICE

To avoid the type of error resulting from confusing the `==` operator with the `=` operator, many C programmers will write an equality expression that contains a variable and a constant, such as `x == 4`, with the constant on the left side and the variable on the right side of the equality operator as in: `4 == x`. This will prevent the logic error that we saw on the previous page from happening when you accidentally replace `==` with `=`.

The reason that this will be caught as a syntax error and the other way around will not is because only a variable can be placed on the left hand side of an assignment expression. Variable names are said to be **lvalues** (for “left values”) because they can be used on the left side of an assignment operator. Constants are said to be **rvalues** (for “right values” because they can be used only on the right side of an assignment operator. Note that lvalues can also be used as rvalues (i.e., variables can appear on either the left or right side of the assignment operator), but rvalues cannot be used as lvalues.



Counter vs. Sentinel Controlled Repetition

- We've seen the three repetition structures in the C language: `while` statement, `do...while` statement, and the `for` statement.
- By now you should be fairly familiar with how each of these statements work. You should also be aware of how the loops are similar in that a `for` statement can basically be replaced by a `while` statement with no loss of functionality.
- As we've seen, the `while` statement and `do...while` statements are more commonly used when the number of repetitions to be made is not known in advance and the `for` statement is more suitable for when the number of repetitions is known in advance.



Counter vs. Sentinel Controlled Repetition

- To further illustrate the selection choice for a repetition statement; consider the following two problems:
- Problem 1: We want the user to enter a set of 10 integer values and our program will compute the sum, product, and average of the ten numbers the user enters.
- Problem 2: We want the user to enter an unknown number of integer values and our program will compute the sum, product, and average of all the numbers entered by the user.
- Considering problem 1 first, we can clearly use any of the repetition statements to help solve this problem. The next three pages show example solutions using all three of these statements; first a `for` statement, and second a `while` statement, and finally a `do...while` statement.

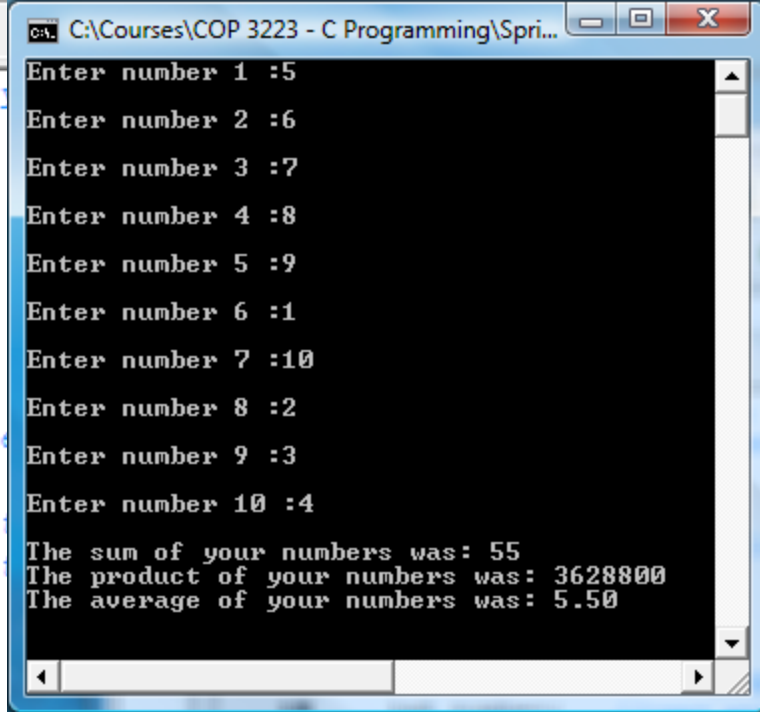



```
2 //January 29, 2009   Written by: Mark Llewellyn
3
4 #include <stdio.h>
5 #define LIMIT 10   //upper limit for loop
6
7 int main()
8 {
9     int counter;   //loop counter
10    int number;    //user entered integer value
11    double sum = 0; //holds sum of integers
12    double product = 1; //holds product of integers
13    double average; //holds average of the integers
14
15    for (counter = 1; counter <= LIMIT; counter++) {
16        printf("Enter number %d :", counter);
17        scanf("%d", &number);
18        printf("\n");
19        sum += number; //add new number
20        product *= number; //multiply new number
21    } //end for stmt
22    printf("The sum of your numbers was: %.0f\n", sum);
23    printf("The product of your numbers was: %.0f\n", product);
24    printf("The average of your numbers was: %.2f\n", sum / LIMIT);
25
26    printf("\n\n");
27    system("PAUSE");
28    return 0;
29 } //end main function
```

```
Enter number 1 :10
Enter number 2 :9
Enter number 3 :8
Enter number 4 :7
Enter number 5 :6
Enter number 6 :5
Enter number 7 :4
Enter number 8 :3
Enter number 9 :2
Enter number 10 :1
The sum of your numbers was: 55
The product of your numbers was: 3628800
The average of your numbers was: 5.50
Press any key to continue . . .
```



```
2 //January 29, 2009   Written by: Mark Llewellyn
3
4 #include <stdio.h>
5 #define LIMIT 10   //upper limit for loop
6
7 int main()
8 {
9     int counter = 1;   //loop counter
10    int number;   //user entered integer value
11    double sum = 0;   //holds sum of integers
12    double product = 1; //holds product of integers
13    double average; //holds average of the integers
14
15    while (counter <= LIMIT) {
16        printf("Enter number %d :", counter);
17        scanf("%d", &number);
18        printf("\n");
19        sum += number;   //add new number
20        product *= number; //multiply new number
21        counter++; //increment counter
22    } //end while stmt
23    printf("The sum of your numbers was: %.0f\n", sum);
24    printf("The product of your numbers was: %.0f\n", product);
25    printf("The average of your numbers was: %.2f\n", sum / LIMIT);
26
27    printf("\n\n");
28    system("PAUSE");
29    return 0;
30 } //end main function
```



```
2 //January 29, 2009   Written by: Mark Llewellyn
3
4 #include <stdio.h>
5 #define LIMIT 10   //upper limit for loop
6
7 int main()
8 {
9     int counter = 1;   //loop counter
10    int number;   //user entered integer value
11    double sum = 0;   //holds sum of integers
12    double product = 1; //holds product of integers
13    double average; //holds average of the integers
14
15    do {
16        printf("Enter number %d :", counter);
17        scanf("%d", &number);
18        printf("\n");
19        sum += number;   //add new number
20        product *= number; //multiply new number
21        counter++; //increment counter
22    } while (counter <= LIMIT); //end do...while stmt
23    printf("The sum of your numbers was: %.0f\n", sum);
24    printf("The product of your numbers was: %.0f\n", product);
25    printf("The average of your numbers was: %.2f\n", sum / LIMIT);
26
27    printf("\n\n");
28    system("PAUSE");
29    return 0;
30 } //end main function
```

```
C:\Courses\COP 3223 - C Programming\Sprin...
Enter number 1 :2
Enter number 2 :4
Enter number 3 :6
Enter number 4 :8
Enter number 5 :10
Enter number 6 :1
Enter number 7 :3
Enter number 8 :5
Enter number 9 :7
Enter number 10 :9
The sum of your numbers was: 55
The product of your numbers was: 3628800
The average of your numbers was: 5.50
```



Counter vs. Sentinel Controlled Repetition

- Now considering problem 2, how do we solve it using these repetition statements (since that's all we have in C) when we do not know in advance how many values the user has to enter.
- Solution 1: Ask the user first to tell your program how many integers they will enter. In this case replace the constant LIMIT and make it an integer variable and read this value first.
- Solution 2: Set aside a special value, called a sentinel, that the user will enter to indicate that they have entered all of the values that they intend to enter.
- A sample program using solution 1 is shown on page 13 and a sample program using solution 2 is shown on page 14.



```
5 #include <stdio.h>
6
7 int main()
8 {
9     int limit; //how many integers the
10    int counter; //loop counter
11    int number; //user entered integer
12    double sum = 0; //holds sum of integers
13    double product = 1; //holds product of integers
14    double average; //holds average of the integers
15
16
17    printf("How many integers will you enter? ");
18    scanf("%d", &limit); printf("\n");
19    for (counter = 1; counter <= limit; counter++) {
20        printf("Enter number %d :", counter);
21        scanf("%d", &number);
22        printf("\n");
23        sum += number; //add new number
24        product *= number; //multiply new number
25    } //end for stmt
26    printf("The sum of your numbers was: %.0f\n", sum);
27    printf("The product of your numbers was: %.0f\n", product);
28    printf("The average of your numbers was: %.2f\n", sum / limit);
29
30    printf("\n\n");
31    system("PAUSE");
32    return 0;
33 } //end main function
```

```
C:\Courses\COP 3223 - C Programming\Sp...
How many integers will you enter? 4
Enter number 1 :5
Enter number 2 :6
Enter number 3 :5
Enter number 4 :7
The sum of your numbers was: 23
The product of your numbers was: 1050
The average of your numbers was: 5.75
Press any key to continue . . .
```



```

7 int main()
8 {
9     int counter = 0; //counts number of
10    int number; //user entered integer
11    double sum = 0; //holds sum of integers
12    double product = 1; //holds product of integers
13    double average; //holds average of the numbers
14
15    printf("Enter an integer (use -999 to stop): ");
16    scanf("%d", &number);
17    while (number != -999) {
18        printf("\n");
19        sum += number; //add new number
20        product *= number; //multiply new number
21        counter++; //increment count of numbers entered
22        printf("Enter an integer (use -999 to stop): ");
23        scanf("%d", &number);
24    } //end while stmt
25    if (counter != 0) {
26        printf("\n");
27        printf("The sum of your numbers was: %.0f\n", sum);
28        printf("The product of your numbers was: %.0f\n", product);
29        printf("The average of your numbers was: %.2f\n", sum / counter);
30    } //end if stmt
31
32    printf("\n\n");
33    system("PAUSE");
34    return 0;
35 } //end main function

```

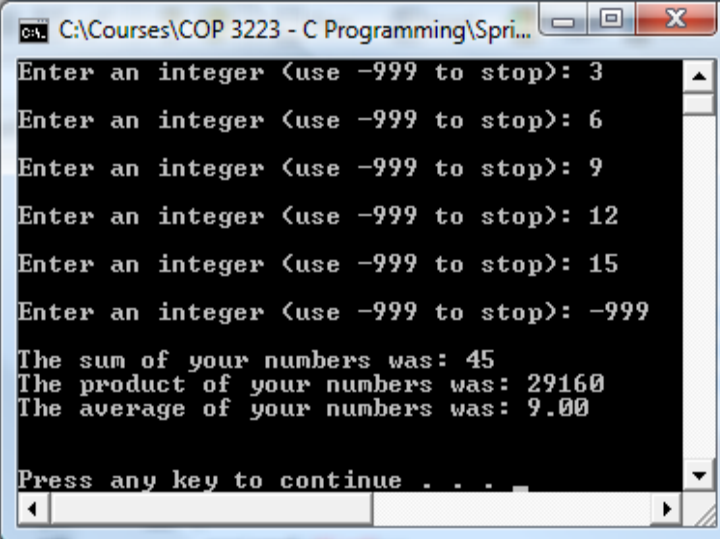
```

C:\Courses\COP 3223 - C Programming\Spring ...
Enter an integer (use -999 to stop): 4
Enter an integer (use -999 to stop): 10
Enter an integer (use -999 to stop): 6
Enter an integer (use -999 to stop): -999
The sum of your numbers was: 20
The product of your numbers was: 240
The average of your numbers was: 6.67
Press any key to continue . . .

```




```
7 int main()
8 {
9     int counter = 0; //counts number of val
10    int number; //user entered integer val
11    double sum = 0; //holds sum of integer
12    double product = 1; //holds product of i
13    double average; //holds average of the i
14
15    printf("Enter an integer (use -999 to st
16    scanf("%d", &number);
17    do {
18        printf("\n");
19        sum += number; //add new number
20        product *= number; //multiply new number
21        counter++; //increment count of numbers entered
22        printf("Enter an integer (use -999 to stop): ");
23        scanf("%d", &number);
24    } while (number != -999); //end do...while stmt
25    if (counter != 0) {
26        printf("\n");
27        printf("The sum of your numbers was: %.0f\n", sum);
28        printf("The product of your numbers was: %.0f\n", product);
29        printf("The average of your numbers was: %.2f\n", sum / counter);
30    } //end if stmt
31
32    printf("\n\n");
33    system("PAUSE");
34    return 0;
35 } //end main function
```

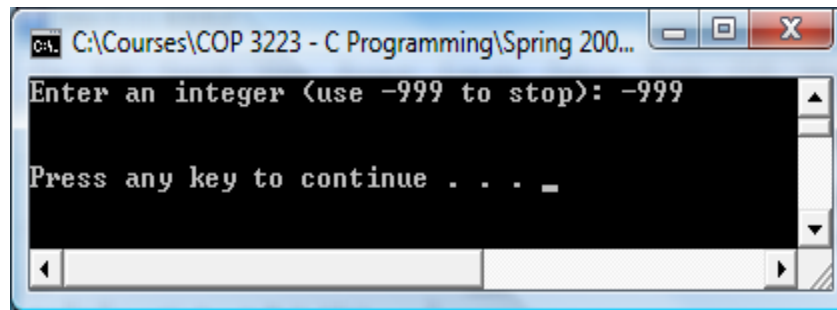


```
C:\Courses\COP 3223 - C Programming\Spr...
Enter an integer (use -999 to stop): 3
Enter an integer (use -999 to stop): 6
Enter an integer (use -999 to stop): 9
Enter an integer (use -999 to stop): 12
Enter an integer (use -999 to stop): 15
Enter an integer (use -999 to stop): -999
The sum of your numbers was: 45
The product of your numbers was: 29160
The average of your numbers was: 9.00
Press any key to continue . . .
```



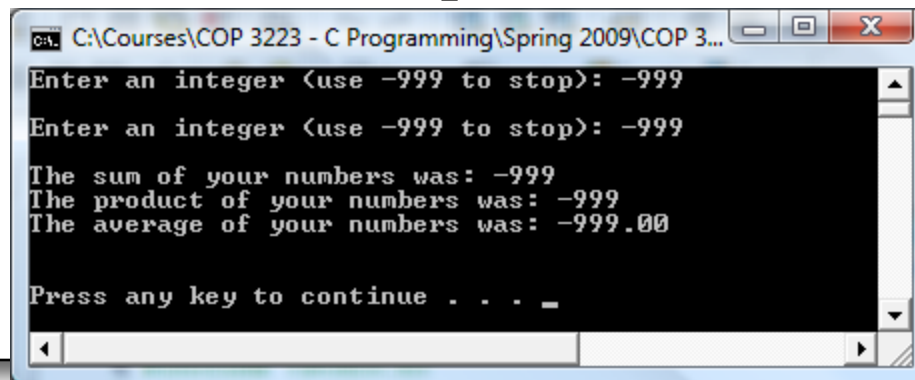
Counter vs. Sentinel Controlled Repetition

- Notice that the sentinel controlled while statement will function properly when the user decides not to enter any numbers at all (i.e., their first value input is -999).



```
C:\Courses\COP 3223 - C Programming\Spring 200...
Enter an integer (use -999 to stop): -999
Press any key to continue . . . _
```

- Will the program on the following page also produce correct results if the user's first input value is -999?



```
C:\Courses\COP 3223 - C Programming\Spring 2009\COP 3...
Enter an integer (use -999 to stop): -999
Enter an integer (use -999 to stop): -999
The sum of your numbers was: -999
The product of your numbers was: -999
The average of your numbers was: -999.00
Press any key to continue . . . _
```



Counter vs. Sentinel Controlled Repetition

- Can you think of a way to use the `do...while` structure from the previous page and still get it to work properly when the user decides not to enter any numbers at all?



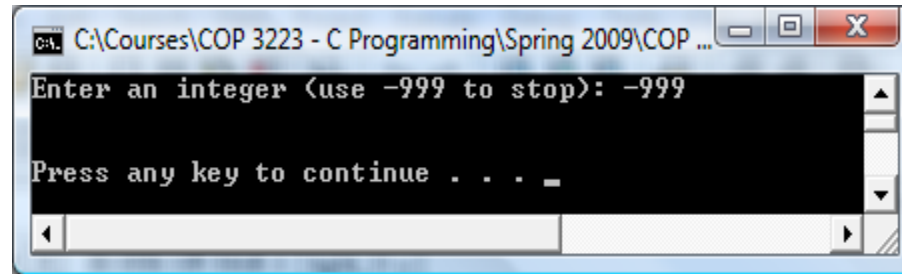
```
7 int main()
8 {
9     int counter = 0; //counts number of values entered
10    int number; //user entered integer value
11    double sum = 0; //holds sum of integers
12    double product = 1; //holds product of integers
13    double average; //holds average of the integers
14
15    printf("Enter an integer (use -999 to stop): ");
16    scanf("%d", &number);
17    do {
18        if (number == -999) {
19            break;
20        } //exit loop if first number was -999
21        printf("\n");
22        sum += number; //add new number
23        product *= number; //multiply new number
24        counter++; //increment count of numbers entered
25        printf("Enter an integer (use -999 to stop): ");
26        scanf("%d", &number);
27    } while (number != -999); //end do...while stmt
28    if (counter != 0) {
29        printf("\n");
30        printf("The sum of your numbers was: %.0f\n", sum);
31        printf("The product of your numbers was: %.0f\n", product);
32        printf("The average of your numbers was: %.2f\n", sum / counter);
33    } //end if stmt
34
35    printf("\n\n");
```

How about using the break statement to exit the loop if the first value was -999?

Will this work?



Counter vs. Sentinel Controlled Repetition

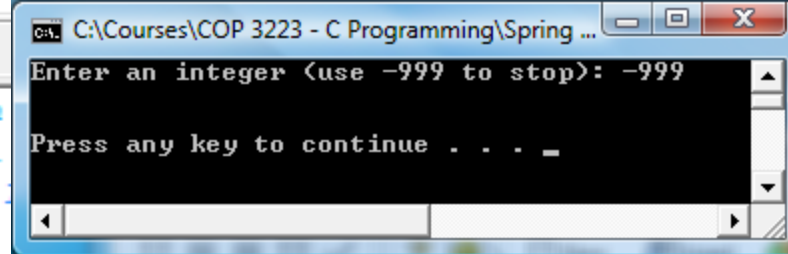


```
C:\Courses\COP 3223 - C Programming\Spring 2009\COP ...
Enter an integer (use -999 to stop): -999
Press any key to continue . . . _
```

- Yes! Remember that a `break` statement when executed inside a `while`, `for`, `do..while`, or `switch` statement, causes an immediate exit from that statement.



```
1 //program to sum, multiply, average an
2 //using while statement and a sentinel
3 //January 29, 2009   Written by: Mark
4
5 #include <stdio.h>
6
7 int main()
8 {
9     int counter = 0; //counts number of values entered
10    int number; //user entered integer value
11    double sum = 0; //holds sum of integers
12    double product = 1; //holds product of integers
13    double average; //holds average of the integers
14
15    printf("Enter an integer (use -999 to stop): ");
16    scanf("%d", &number);
17    do {
18        if (number != -999)
19            { printf("\n");
20              sum += number; //add new number
21              product *= number; //multiply new number
22              counter++; //increment count of numbers entered
23              printf("Enter an integer (use -999 to stop): ");
24              scanf("%d", &number);
25              } //end if
26    } while (number != -999); //end do...while stmt
27    if (counter != 0) {
28        printf("\n");
29        printf("The sum of your numbers was: %.0f\n", sum);
```



```
C:\Courses\COP 3223 - C Programming\Spring ...
Enter an integer (use -999 to stop): -999
Press any key to continue . . . .
```

This also works!



The `continue` Statement

- There is also another statement in C called the `continue` statement that will work in this case.
- The `continue` statement, when executed in a `while`, `for` or `do...while` statement, skips the remaining statements in the body of that control statement and performs the next iteration of the loop.
 - In `while` and `do...while` statements, the loop-continuation conditional expression is evaluated immediately after the `continue` statement is executed.
 - In a `for` statement, the increment expression is executed, then the loop continuation conditional expression is evaluated.



```
7 int main()
8 {
9     int counter = 0; //counts number of va
10    int number; //user entered integer va
11    double sum = 0; //holds sum of integers
12    double product = 1; //holds product of integers
13    double average; //holds average of the integers
14
15    printf("Enter an integer (use -999 to stop): ");
16    scanf("%d", &number);
17    do {
18        if (number == -999) {
19            continue;
20        } //exit loop if first number was -999
21        printf("\n");
22        sum += number; //add new number
23        product *= number; //multiply new number
24        counter++; //increment count of numbers entered
25        printf("Enter an integer (use -999 to stop): ");
26        scanf("%d", &number);
27    } while (number != -999); //end do...while stmt
28    if (counter != 0) {
29        printf("\n");
30        printf("The sum of your numbers was: %.0f\n", sum);
31        printf("The product of your numbers was: %.0f\n", product);
32        printf("The average of your numbers was: %.2f\n", sum / counter);
33    } //end if stmt
34
35    printf("\n\n");
```

Enter an integer (use -999 to stop): -999

Press any key to continue . . . _



Practice Problems

1. Find the error in each of the following code segments and explain how to correct it.

(a)

```
int x = 1;
while (x <= 10); {
    printf("x is: %d\n", x);
    x++;
}
```

(b)

```
//This code should print the
//values 1 to 10.int x = 1;
int n = 1;
while (n < 10){
    printf("%d\n", x++);
}
```

```
int n;
switch (n) {
    case 1: printf("Number is 1\n");
    case 2: printf("Number is 2\n");
            break;
    case 3: printf("Number is 3\n");
    default:
        printf("Number is not 1..3\n");
        break;
}
```

(c)



Practice Problems

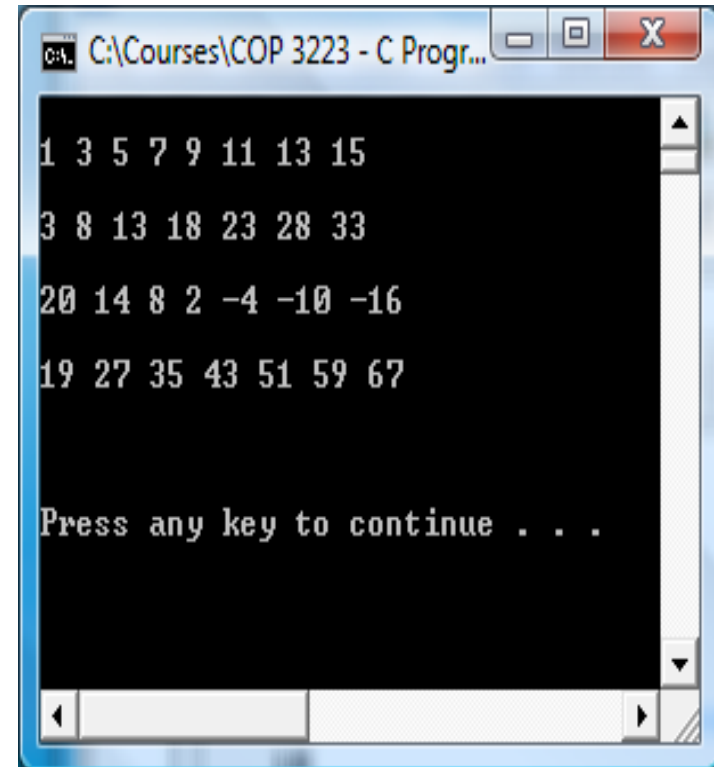
2. Construct a C program that uses `for` loops to produce the following sequences of values:

1, 3, 5, 7, 9, 11, 13, 15

3, 8, 13, 18, 23, 28, 33

20, 14, 8, 2, -4, -10, -16

19, 27, 35, 43, 51, 59, 67



```
C:\Courses\COP 3223 - C Progr...  
1 3 5 7 9 11 13 15  
3 8 13 18 23 28 33  
20 14 8 2 -4 -10 -16  
19 27 35 43 51 59 67  
  
Press any key to continue . . .
```



Practice Problems

3. What does the following program do?

```
control Structures Revisited Practice Prob 2.c | Control Structures Revisited Practice Prob 3.c |
1 //Control Structures Revisited Practice Problem 3
2 //January 29, 2009   Written by: Mark Llewellyn
3
4 #include <stdio.h>
5
6 int main()
7 {
8     int x, y, i, j;
9
10    printf("Enter two integers in the range 1-20: ");
11    scanf("%d%d", &x, &y);
12
13    for (i = 1; i <= y; i++) {
14        for (j = 1; j <= x; j++) {
15            printf("@ ");
16        }
17        printf("\n");
18    }
19
20    printf("\n\n");
21    system("PAUSE");
22    return 0;
23 }//end main function
```



Practice Problems

4. Construct a C program that produces the following output.

```

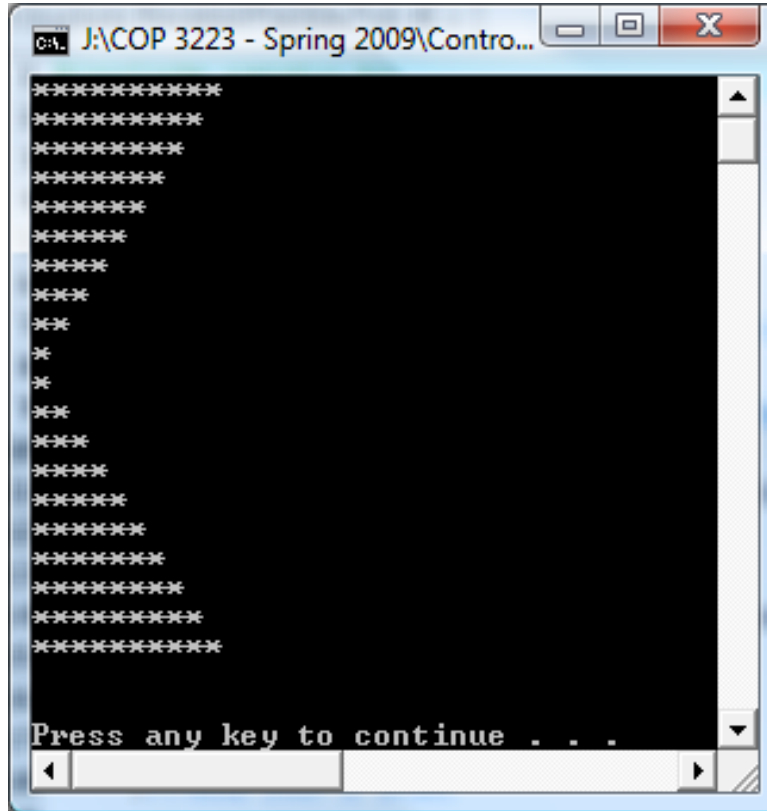
C:\Courses\COP ...
*
**
***
****
*****
*****
****
***
**
*
*
**
***
****
*****
*****
****
***
**
*

```



Practice Problems

7. Construct a C program that produces the following output.



```
J:\COP 3223 - Spring 2009\Contro...
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
Press any key to continue . . .
```

