East West University
Department of Computer Science & Engineering
Plot No-A/2, Main Road, Jahurul Islam City, Aftabnagar, Dhaka-1219

Lab Manual
Course : CSE -105
Credit Title : Structured Programming
Instructor : Md.Shamsujjoha (MSJ)

Lab-6: Basic Program Control: Loops Continues (***for***) …

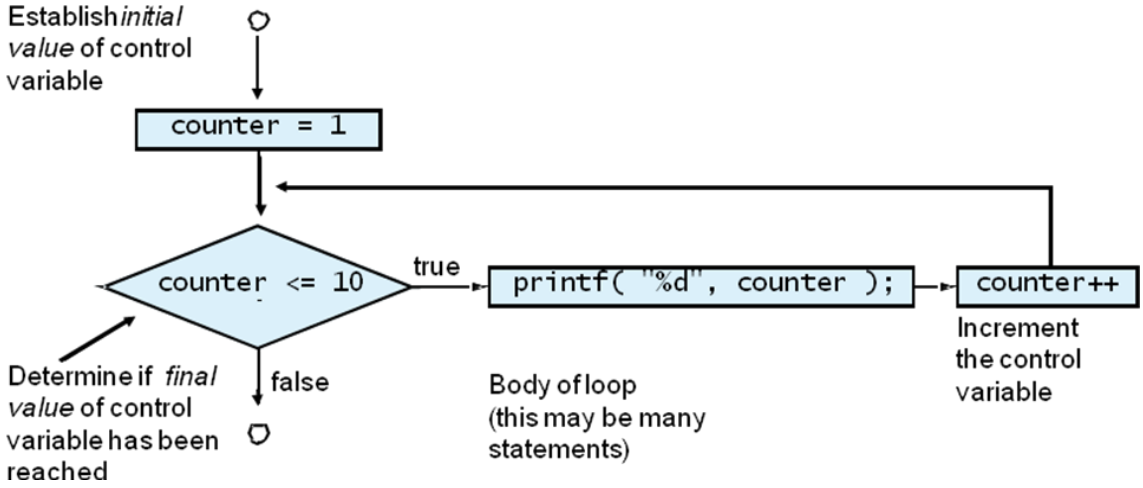
In the previous two labs we go through the detail of C's while loop and do while loop. In this lab we will have a look at C's ***for*** loop. The ***for*** statement is a C programming construct that executes a block of one or more statements a certain number of times. It is sometimes called the ***for*** loop because program execution typically loops through the statement more than once. You've seen a few for statements used in programming examples earlier in the class. Now you're ready to see how actually the ***for*** statement works. A for statement has the following structure:

```
for(initialization; condition; increment/decrement)
    statement;
```

initialization, condition, and increment/decrement are all C expressions, and statement is a single or compound C statement. When a for statement is encountered during program execution, the following events occur:

1. The expression initialization is evaluated. initialization is usually an assignment statement that sets a variable to a particular value.
2. The expression condition is evaluated. condition is usually a relational expression.
3. If condition evaluates to false (that is, as zero), the for statement terminates, and execution passes to the first statement following statement.
4. If condition evaluates to true (nonzero), the C statement in statement are executed.
5. The expression increment is evaluated, and execution returns to step 2.

Look at the following flowchart:



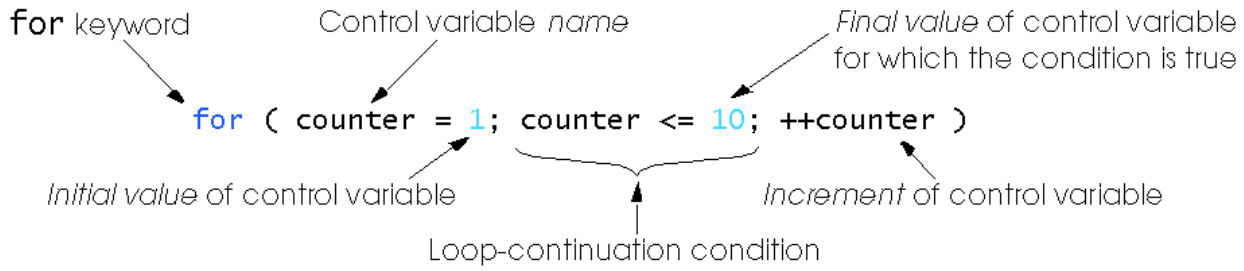
Above flowchart will print 1 2 3 up to 10. We can code the above flowchart without the loop. The corresponding C code of the above flowchart without the loop is as follows,

```
int counter = 1;
printf("%d ",counter); //1

counter = counter + 1 ;
printf("%d ",counter);  //2


counter = counter + 1 ;
printf("%d ",counter); //3

… … … … … … … …
… … … … … … … …
… … … … … … … …

counter = counter + 1 ;
printf("%d ",counter); //10
```

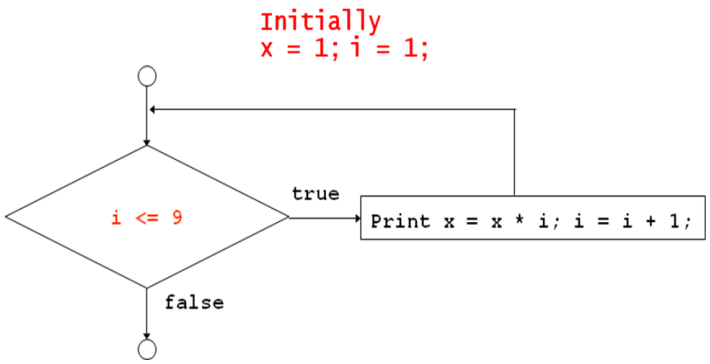
Same thing can be done with a single line of for loop followed by a printf functions, which is shown below,

```
for(int counter = 1; counter<=10; counter = counter+1 )
printf("%d ",counter);
```

Here,



In this loop example, if you want to print 1 to 100, you just need to change one thing i.e., `counter<=10` to `counter<=100.` On the other hand, if you code the above flowchart without loop, you need 90 more printf and increment of counter variable.



Exercise 1: Your 1st task is to write a C programme with a single for loop followed by a single printf function of the flowchart shown in left.

Q.1: Can you code this flowchart without any loop (for or while)?

Nesting for Statements, Just like the if and while statements, for statements can also be nested. Following shows an example of nested for statements.

```
for (i=1; i<=5; i++) {
      for (j=1; j<=4; j++){
          printf("*");
            }
      printf("\n");
}
```

Write this above code inside a main function, compile and run. The output should be is as follows :

```
****
****
****
****
****
```

Exercise 2: Modify the for loop of Exercise 3 such that the output be as follows, you are allowed only to modify the for loops of exercise 3. (use only for loops)

```
*****
****
***
**
*
```

Exercise 3: Modify the for loop of Exercise 4 and add this printf(" "); function in your code of Exercise 4 such that your output be as follows. (use only for loops)

```
    *
   * *
  * * *
 * * * *
* * * * *
```

Exercise 4: Write a program with for loops, printf and scanf functions only such that, will take an integer number `***n***` as input and print the result of the following series: (use only for loops)

a. 1+2+3+. . .+***n***
b. 1+1/2+1/3+. . . + ***n***
c. 1.2+3.4+5.6+. . .+(***n***-1).***n***
d. 1.2.3+2.3.4+3.4.5+. . .+***n***.(***n***+1).(***n***+2)
e. 1+2+4+7+11+ . . . (up to `***n***` numbers)

Sample Input	Sample Output
4	a) 10 b) 2.08 c) 14 d) 210 e) 14

Exercise 5: Write a program that prints in two columns ***n*** even numbers starting from 2, and a running sum of those values. For example suppose user enters 5 for ***n***, then the program should generate the following table: (use only for loops)

Enter n (the number of even numbers): 5

Value	Sum
2	2
4	6
6	12
8	20
10	30

Exercise 6 : Write a program that will take a number `***n***` as input and print `***n***` lines of output according to the following sample output, use only for loops.

Sample Input	Sample Output
3	1 2 3 4 5 6
5	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Exercise 7 : Solve the ACM problem 11777 and 100 using only for loops. The detailed problem are attached at the end. This exercise carries 60% of today's lab points.

Reminder: **Lab 3-to-7 Viva → 22/11/2012 Time : 8.00-1.00 .**
Submission Date: 21/11/2012
Submit Hard Copy (Hand Written/printed) of Home Works, Exercises & ACM Problems also Submit Soft Copy (Source Code)

Thanks
Maheen Islam, Dr. Taskeed Jabid and Md. Shamsujjoha
Faculty members, Department of Computer Science & Engineering, East West University
Email: maheen@ewubd.edu, tasked@ewubd.edu and dishacse@yahoo.com

Problem G
Automate the Grades

Input: Standard Input
Output: Standard Output

The teachers of "Anguri Begam Uccha Biddalya", a school located in the western region of Sylhet, currently follows a manual system for grading their students. The manual process is very time consuming and error prone. From the next semester they have decided to purchase some computers so that the whole grading process can be automated. And yes, you guessed it - they have hired you to write a program that will do the job.

The grading of each course is based on the following weighted scale:

- Term 1 – 20%
- Term 2 – 20%
- Final – 30%
- Attendance – 10%
- Class Tests – 20%

The letter grades are given based on the total marks obtained by a student and is shown below:

- `A >= 90%`
- `B >= 80% & < 90%`
- `C >= 70% & < 80%`
- `D >= 60% & < 70%`
- `F  < 60%`

Term 1 and Term 2 exams are out of 20 each, *Final* is out of 30 and *Attendance* given is out of 10. Three class tests are taken per semester and the average of best two is counted towards the final grade. Every class test is out of 20.

Example: Say Tara obtained marks of 15, 18, 25 and 8 in *Term 1*, *Term 2*, *Final* and *Attendance* respectively. Her 3 class test marks are 15, 12 and 17. Since average of best 2 will be counted, her *class test* mark will be equal to (15 + 17) / 2 = 16. Therefore, total marks = 15 + 18 + 25 + 8 + 16 = 82 and she will be getting a B.

Input
The first line of input is an integer **T**(**T**<100) that indicates the number of test cases. Each case contains 7 integers on a line in the order ***Term1 Term2 Final Attendance Class_Test1 Class_Test2 Class_Test3***. All these integers will be in the range [0, total marks possible for that test].

Output
For each case, output the case number first followed by the letter grade {A B C D F}. Follow the sample for exact format.

Sample Input	Output for Sample Input
3 15 18 25 8 15 17 12 20 20 30 10 20 20 20 20 20 30 10 18 0 0	Case 1: B Case 2: A Case 3: B

Problemsetter: Sohel Hafiz
Special Thanks to: Shamim Hafiz

100 The $3n + 1$ problem

Background

roblems in Computer Science are often classified as belonging to a certain class of problems (e.g., N , Unsolvable, Recursive). In this problem you will be analyzing a property of an algorithm whose classification is not known for all possible inputs.

The roblem

Consider the following algorithm:

1. input n
2. print n
3. if $n = 1$ then STO
4. if n is odd then $n \longleftarrow 3n + 1$
5. else $n \longleftarrow n/2$
6. GOTO 2

Given the input 22, the following sequence of numbers will be printed

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

It is conjectured that the algorithm above will terminate (when a 1 is printed) for any integral input value. Despite the simplicity of the algorithm, it is unknown whether this conjecture is true. It has been verified, however, for all integers n such that $0 < n < 1,000,000$ (and, in fact, for many more numbers than this.)

Given an input n, it is possible to determine the number of numbers printed before and including the 1 is printed. For a given n this is called the *cycle-length* of n. In the example above, the cycle length of 22 is 16.

For any two numbers i and j you are to determine the maximum cycle length over all numbers between and including both i and j.

The Input

The input will consist of a series of pairs of integers i and j, one pair of integers per line. ll integers will be less than 10,000 and greater than 0.

You should process all pairs of integers and for each pair determine the maximum cycle length over all integers between and including i and j.

The Output

For each pair of input integers i and j you should output i, j, and the maximum cycle length for integers between and including i and j. These three numbers should be separated by at least one space with all three numbers on one line and with one line of output for each line of input. The integers i and j must appear in the output in the same order in which they appeared in the input and should be followed by the maximum cycle length (on the same line).

Sample Input

```
1 10
100 200
201 210
900 1000
```

Sample Output

```
1 10 20
100 200 125
201 210 89
900 1000 174
```