FEU-EAST ASIA COLLEGE
COLLEGE OF ENGINEERING • COLLEGE OF COMPUTER STUDIES

INFORMATION TECHNOLOGY EDUCATION DEPARTMENT

ITPROG1

(INTRODUCTION TO PROGRAMMING)

EXERCISE

5

REPETITION CONTROL STRUCTURE

score

Crystel Anne Del Mundo Name of Student	Prof. Cruz Name of Professor
October 27, 2016 Date Performed	October 27, 2016 Date Submitted

OBJECTIVES

At the end of this exercise, students must be able to:

- Know the loop structure
- Know the different ways on how the loop is controlled
- Simulate a loop statement to determine the output

IV. BACKGROUND INFORMATION

- Loop Structure condition can be tested first or tested later
- Loops can be controlled by condition, counter, or sentinel

For loop

- Condition is tested first
- Loop is controlled by a counter
- Syntaxes

```
for (initial value ; condition; update counter)
        statement;
Or
        for (initial value ; condition; update counter) {
            statement;
            statement;
     }
```

While loop

- Condition is tested first
- Loop is controlled by condition or a counter
- Syntax

```
   while (condition)
        statement;
 Or
        while (condition) {
            statement;
            statement;
        }
```

Do-while loop

- Statements in the loop are executed first (at least once, and condition is tested last
- Loop is controlled by a condition or counter
- Syntax

```
do {
      statement;
      statement;
} while (condition);
```

- statement;

V. LABORATORY ACTIVITY

ACTIVITY 5.1: Product of two numbers using while loop

Calculate the product of two numbers without using the multiiplication (*) operator. Consider the sample output.
Enter 1st number: 2
Enter 2nd number: 2
Product: 4

```
/*
 * C++ Program to multiply numbers without * operator
 */
#include <iostream>

int main()
{
   long op1, op2, product = 0;

   std::cout << "Enter the numbers ";
   std::cin >> op1 >> op2;
   while(op2 > 0)
   {
      product = product + op1;
      op2--;
   }
   std::cout << std::endl << "The product of " << op1
         << " and " << op2 << " is " << product;
}
```

ACTIVITY 5.2: Even and Odd numbers using do-while loop

Using do-while loop, write a program that will ask the user to input an integer number. Display all even and odd numbers from 1 to the user's input. Consider the sample output.
Output:

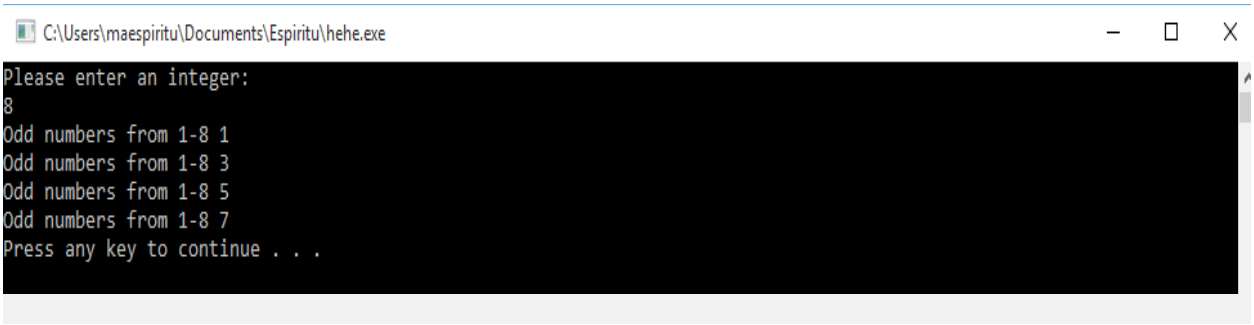
Enter an integer number: 10
Odd numbers from 1 - 10: 1 3 5 7 9
Even Numbers from 1 - 10: 2 4 6 8 10

```
#include <iostream>
using namespace std;

int main ()
{
int num;
int odd=1;
int even=2;
cout << "Please enter an integer: " << endl;
cin >> num;

do
{
cout << "Odd numbers from 1-" << num << " " << odd << endl;
odd=odd+2;
}while (odd<num);
system ("pause");
return 0;

}
```



ACTIVITY 5.3: Determining highest and lowest value

Using do-while loop, Write a program that will ask for five (5) integer numbers. Determine the highest and lowest value.
Consider the sample output.
Output:

Enter 1st number: 5
Enter 2nd number: 10
Enter 3rd number: 2
Enter 4th number: 50
Enter 5th number: 30

```
      The highest value is: 50
      The lowest value is: 2
int count;
int largest;
int smallest;

cout << "Enter 5 integer values ." << endl;

for (count = 0; count < SIZE; count++)
{
   cout << "\nEnter an integer value: ";
   cin  >> values[count];
}

largest = smallest = values[0];
for (count = 1; count < SIZE; count++)
{
   if (values[count] > largest)
      largest = values[count];
   if (values[count] < smallest)
      smallest = values[count];
}


cout << "\nThe largest value entered is " << largest << endl;
cout << "The smallest value entered is " << smallest << endl << endl;


 system("pause");
  return 0;
}
```

ACTIVITY 5.4: Getting the factorial

Using for loop, Write a program that will ask the user to input an integer number. Compute for the factorial value. Factorial is the product of all positive integers less than or equal to n.
Example: 5! = 5*4*3*2*1 = 120
Consider the sample output:
Enter an integer number: 5
The factorial of 5 is: 120

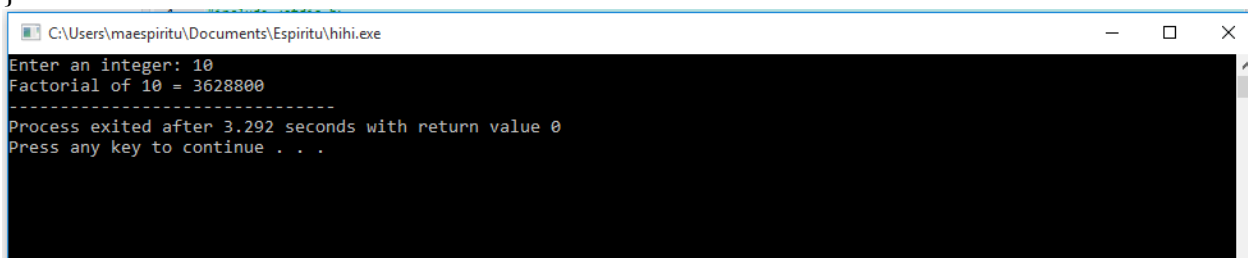
```
# include <stdio.h>
int main()
{
   int n, i;
   unsigned long long factorial = 1;

   printf("Enter an integer: ");
   scanf("%d",&n);

   // show error if the user enters a negative integer
   if (n < 0)
      printf("Error! Factorial of a negative number doesn't exist.");

   else
   {
      for(i=1; i<=n; ++i)
      {
         factorial *= i;              // factorial = factorial*i;
      }
      printf("Factorial of %d = %llu", n, factorial);
   }

   return 0;
}
```



ACTIVITY 5.5: Printing stars

Using nested loop, write a program that will ask for an integer number. Based from the input display the following output.
Example:
Enter a number: 5
*

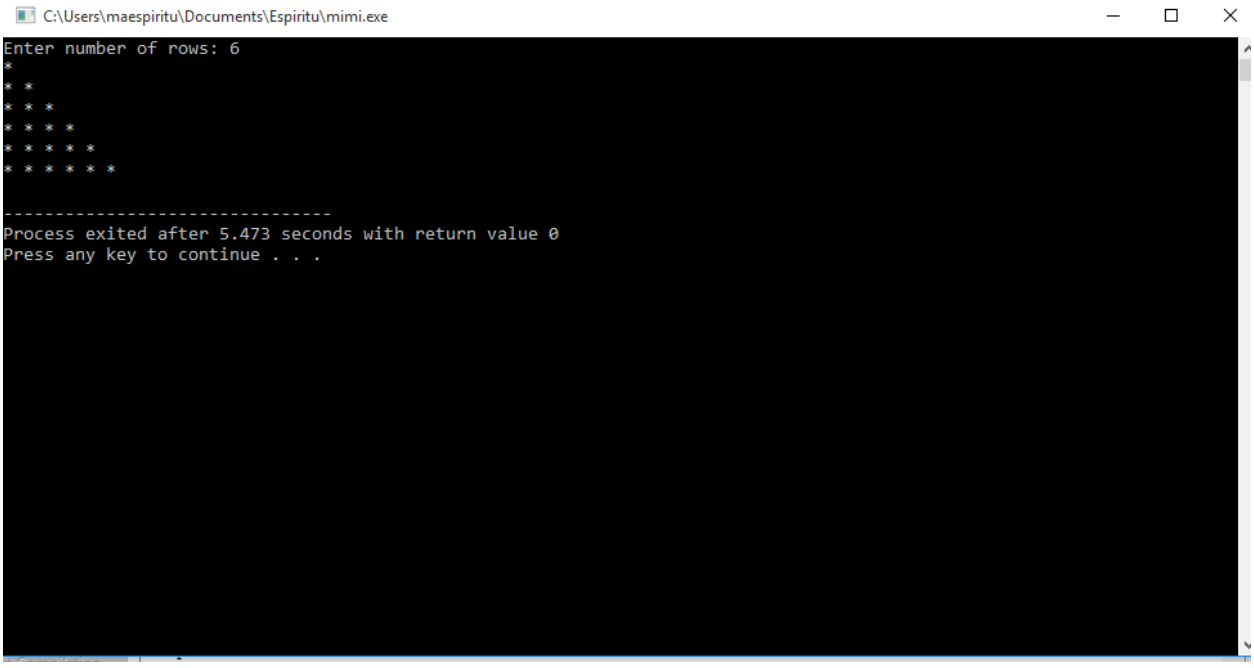
**
***
****

```
#include <iostream>
using namespace std;

int main()
{
   int rows;

   cout << "Enter number of rows: ";
   cin >> rows;

   for(int i = 1; i <= rows; ++i)
   {
      for(int j = 1; j <= i; ++j)
      {
         cout << "* ";
      }
      cout << "\n";
   }
   return 0;
}
```



VI. QUESTION AND ANSWER

1. What is the significance of using repetition control structures?

The significance of using repetition control structures is to execute a block of code a number of times based on some condition. In the world of programming, we generally will use two types of repetition control structures in counting and conditional. We commonly refer to repetition control structures as loops.

2. Which is the easiest way in looping? Explain.

Break and continue, this is for me the easiest way, you can use this loop to break a statement at any time. This can be very useful if you want to stop running a loop because a condition has been met other than the loop end condition.

V. ASSESSMENT

Department	Information Technology
Subject Code	ITPROG1
Description	INTRODUCTION TO PROGRAMMING
Term/Academic Year	

Note: The following rubrics/metrics will be used to grade students' output in the lab Exercise 5.

Program (100 pts.)	**(Excellent)**	**(Good)**	**(Fair)**	**(Poor)**
Program execution (20pts)	Program executes correctly with no syntax or runtime errors **(18-20pts)**	Program executes with less than 3 errors **(15-17pts)**	Program executes with more than 3 errors **(12-14pts)**	Program does not execute **(10-11pts)**
Correct output (20pts)	Program displays correct output with no errors **(18-20pts)**	Output has minor errors **(15-17pts)**	Output has multiple errors **(12-14pts)**	Output is incorrect **(10-11pts)**
Design of output (10pts)	Program displays more than expected **(10pts)**	Program displays minimally expected output **(8-9pts)**	Program does not display the required output (**6-7pts)**	Output is poorly designed **(5pts)**
Design of logic (20pts)	Program is logically well designed **(18-20pts)**	Program has slight logic errors that do no significantly affect the results **(15-17pts)**	Program has significant logic errors **(3-5pts)**	Program is incorrect **(10-11pts)**
Standards (20pts)	Program code is stylistically well designed **(18-20pts)**	Few inappropriate design choices (i.e. poor variable names, improper indentation) **(15-17pts)**	Several inappropriate design choices (i.e. poor variable names, improper indentation) **(12-14pts)**	Program is poorly written **(10-11pts)**
Delivery	The program was	The program was	The program was	The program was

(10pts)	delivered on time. **(10pts)**	delivered a day after the deadline. **(8-9pts)**	delivered two days after the deadline. **(6-7pts)**	delivered more than two days after the deadline. **(5pts)**

Topic	REPETITION CONTROL STRUCTURE
Lab Activity No	5.1
Lab Activity	Product of two numbers using while loop
CLO	2, 3
Program execution (20)	
Correct output (20)	
Design of output (10)	
Design of logic (20)	
Standards (20)	
Delivery (10)	
TOTAL	

Topic	REPETITION CONTROL STRUCTURE
Lab Activity No	5.2
Lab Activity	Even and Odd numbers using do-while loop
CLO	2, 3
Program execution (20)	
Correct output (20)	
Design of output (10)	
Design of logic (20)	
Standards (20)	
Delivery (10)	
TOTAL	

Topic	REPETITION CONTROL STRUCTURE
Lab Activity No	5.3
Lab Activity	Determining highest and lowest value
CLO	2, 3
Program execution (20)	
Correct output (20)	
Design of output (10)	
Design of logic (20)	

Standards (20)	
Delivery (10)	
TOTAL	

Topic	REPETITION CONTROL STRUCTURE
Lab Activity No	5.4
Lab Activity	Getting the factorial
CLO	2, 3
Program execution (20)	
Correct output (20)	
Design of output (10)	
Design of logic (20)	
Standards (20)	
Delivery (10)	
TOTAL	

Topic	REPETITION CONTROL STRUCTURE
Lab Activity No	5.5
Lab Activity	Printing stars
CLO	2, 3
Program execution (20)	
Correct output (20)	
Design of output (10)	
Design of logic (20)	
Standards (20)	
Delivery (10)	
TOTAL	