Assignment 3 - Repetition and Functions

Repetition Structures

1.1 Differentiate between a “while” and a “do-while” loop.

SOL.

while loop

- In while loop, condition is evaluated first and then the statements inside the loop body get executed.
- Syntax:
  while(condition)
  {
  statements(s);
  }

do- while loop

- In do-while loop, statements inside do-while gets executed first and then the condition is evaluated
- Syntax:
  do
  {
  statements(s);
  } while(condition);

1.2 Predict the output for each of the following:

i)

```
int number = 1;
do
{
    cout << 2*number;
    number = number + 1;
} while (number <= 3);
```

OUTPUT: 246

ii) int number = 1;
while (number <= 3)
{
cout << 2*number;
number = number + 1;
}

OUTPUT: 246

1.3 The following program fragment is supposed to input numbers from the user and display them until a 0 is entered, but a statement is missing. What is the missing statement and where should it be inserted?

```
cin >> number;
while (number != 0)
{
cout << number;
}
```

SOL.

```
cin >> number;
while (number!= 0)
{
cout << number;
cin >> number;
}
```

1.4 Replace the while loop in the Counting program and the Summation program with the do-while loop.

SOL.

```
//Counting Program

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

```
int main()
{
int marks, passing_marks;
int counter = 0; //initialize counter to zero
cout <<"This program counts the number of students that have passed."<< endl;
cout <<"Enter the passing marks: ";
cin >> passing_marks;
cout <<"Enter a student's marks(Enter a negative number to quit.): ";
cin >> marks;
do{
if (marks >= passing_marks)
counter = counter + 1; // increment counter if marks are more than the
                       //passing_marks
cout <<"Enter a student's marks(Enter a negative number to quit.): ";
cin >> marks;
} while (marks != -1); // loop will exit when sentinel (-1) is entered
cout << endl<<endl;
cout << "Number of students that have passed are " << counter;
getchar();
return 0;
}
```

```
//Summation program

#include <iostream>
using namespace std;
int main()
{
int marks;
int accumulator = 0; //initialize accumulator to zero
cout <<"This program sums the marks of students."<< endl;
cout <<"Enter a student's marks(Enter a negative number to quit.): ";
cin >> marks;
do{
accumulator = accumulator + marks; // increment accumulator by
//input marks
cout <<"Enter a student's marks(Enter a negative number to quit.): ";
cin >> marks;
} while (marks != -1);  // loop will exit when sentinel (-1) is entered

cout << endl<<endl;
```

```
cout << "Sum of marks scored by students is: " << accumulator;
getchar();
return 0;
}
```

1.5 Write a program to count the number of students when a set of student marks are entered as input.

SOL.

```
#include <iostream>
using namespace std;
int main()
{
int marks;
int stud_count = 0;   //initialize stud_count to zero
cout <<"Enter a student's marks(Enter a negative number to quit.): ";
cin >> marks;
do{
stud_count++;
cout <<"Enter a student's marks(Enter a negative number to quit.): ";
cin >> marks;
} while (marks != -1);  // loop will exit when sentinel (-1) is entered

cout << endl<<endl;
cout << "Total number of students are: " << stud_count;
getchar();
return 0;
}
```

1.6 Write a program to find the average marks scored by a set of students.

SOL.

```
#include <iostream>
using namespace std;
int main()
{
int marks,avg_marks;
int stud_count = 0;   //initialize stud_count to zero
int accumulator = 0;          //initialize accumulator to zero
cout <<"Enter a student's marks(Enter a negative number to quit.): ";
cin >> marks;
```

```
do{
stud_count++;
accumulator = accumulator + marks; // increment accumulator by
                                   //input marks
cout <<"Enter a student's marks(Enter a negative number to quit.): ";
cin >> marks;
} while (marks != -1);  // loop will exit when sentinel (-1) is entered
avg_marks = accumulator/stud_count;
cout << endl<<endl;
cout << "Average marks of students are: " << stud_count;
getchar();
return 0;
}
```

2.1 Define a counter controlled loop with an example.

SOL.

A counter controlled loop is also known as definite repetition loop,since the number of iterations is known before the loop begins to execute.
It has the following components:

1. A control variable
2. The increment or decrement command to modify control variable
3. The loop terminating condition

For example:

```
//program to print first five natural numbers
#include<iostream>
using namespace std;
int main()
{
for(int i=1;i<=5;++i)
cout<<i;
return 0;
}
```

2.2 Predict the output:

1. for (int count = 10; count < 6; count = count -2)
cout << count << endl;

OUTPUT: NO OUTPUT

2. for (int count = 0; count > 6; count--)
cout << count << endl;

OUTPUT: NO OUTPUT

2.3 Write a program that prints multiples of 10 upto 100.

SOL.

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

        int main()
        {
                for(int i=10;i<=100;i+=10)
                cout<<i<<endl;
                return 0;
        }
```

2.4 Write a program that prints the following pattern:

55555
4444
333
22
1

SOL.

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

        int main()
        {
                for(int i=5;i>=1;--i)
                {
                        for(int j=i;j>=1;j--)
                        {       cout<<i;        }
                        cout<<endl;
                }
                return 0;
        }
```

3.1 What are the differences between break, continue and return?

SOL.

These are jumping statements.

Break is used to exit from a loop or a switch-case.
Continue is used to move the control to the next iteration of the loop.
Return is used to return a value from a function.

3.2 Predict the output

- **for (i = 10; i > 0; i--)**

```
{
    cout << i << endl;

    if (i == 4)
        break;
}
cout << "done"<< endl;
```

OUTPUT:

```
10
9
8
7
6
5
Done
```

- **for (int n=10; n>0; n--)**

```
{
if (n==5) continue;
cout << n << ", ";
}
cout << "done" << endl;
```

OUTPUT: 10,9,8,7,6,4,3,2,1,done

3.3 Write a program that accepts a character input and validates that the character is only between a to z.

SOL.

```
#include<iostream>
using namespace std;
int main()
{
        char ch;
        cout<<”enter a character : ”;
        cin>>ch;
        if(ch >= ’a’ && ch <= ‘z’)
        cout<<”character is between a to z”;
        else cout<<”character is not between a to z”;
        return 0;
}
```

3.4 Write a program to find the second largest number from a list of numbers entered as input.

SOL.

```
#include<iostream>
using namespace std;
int main()
 {
   int num,largest=0,sec_largest=0;
   cout<<"Enter a number (-1 to stop) : ";
   cin>>num;
   do{
     if(num > largest )
     {
       sec_largest=largest;
       largest=num;
     }
     else if(num > sec_largest)
     {
       sec_largest = num;
     }
     cout<<"Enter a number (-1 to stop) : ";
     cin>>num;
```

```
   }while(num != -1);
   cout<<"Second largest numbers is : "<<sec_largest;
   return 0;
 }
```

Output :

```
Enter a number (-1 to stop) : 4
Enter a number (-1 to stop) : 6
Enter a number (-1 to stop) : 9
Enter a number (-1 to stop) : -1
Second largest numbers is : 9
```

3.5 Write a program to compute C(n,m) where C(n,m) = n! / (m!) * (n-m)! given n and m as input.

SOL.

```
#include<iostream>
using namespace std;
int main()
{
int n,m;
int num1 = 1, num2 =1, num3 = 1;
cout<<"enter the value of n and m";
cin>>n>>m;
for(int i=n;i>=1;i--)
{       num1=num1*i;        }
for(int j=m;j>=1;j--)
{       num2 = num2*j;      }
for(int k=(n-m);k>=1;k--)
{       num3 = num3*k;      }
int c;
c=(num1)/((num2)*(num3));
cout<<"c(n,m) = "<<c;
return 0;
}
```

Functions

1.1 What is the return type of the function with prototype: "int func(char a, float b, double c);"
A. char
B. int
C. float
D. double
SOL. B. int

1.2 Which of the following is a valid function call (assuming the function exists)?
A. abc;
B. abc a, b;
C. abc();
D. int abc();
SOL. C. abc();

1.3 Which of the following is a complete function?
A. int qaz();
B. int qaz(int x) {return x=x+1;}
C. void qaz(int) {cout<<"Hello"}
D. void qaz(x) {cout<<"Hello"}
SOL. B. int qaz(int x) {return x=x+1;}

1.4 Write a C++ program which uses a function to find sum of numbers between two given numbers. These two numbers are passed to the function and the sum is returned from it.
SOL.

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

```
float sumoftwo (float a,float b)
        {       return a+b;  }
 int main()
 {
        float x,y,sum;
        cout<<"Enter two numbers : ";
        cin>>x>>y;
        sum = sumoftwo(x,y);
        cout<<"Sum : "<<sum;
        return 0;
}
```

Output: Enter two numbers : 2
8
Sum : 10

1.5 Write a C++ program that invokes a function eql() to find whether four numbers a, b, c, d passed to eql() satisfy the equation a3 + b3 +c3 = d3 or not. eql() should return 0 if the above equation is satisfied with the passed numbers otherwise it returns -1.

SOL.

```
#include<iostream>
#include<cmath>
using namespace std;
 int eql (double a,double b,double c,double d)
 {
  if((pow(a,3)+pow(b,3)+pow(c,3)) == pow(d,3))
return 0;
else return -1;
}
 int main()
 {
   double a,b,c,d;
   cout<<"enter four numbers \n ";
    cin>>a>>b>>c>>d;
    return eql(a,b,c,d);
  }
```

1.6 In what conditions will you use inline functions?

SOL. I will use inline functions when the function is small because this overhead occurs for small functions because execution time of small function is less than the switching time. inline functions are used to reduce the function call overhead.

1.7 How is an inline function different from a preprocessor macro?

SOL. Inline functions are used to reduce the function call overhead.
However macro definitions are another solution to the problem of functions taking a lot of time.
Inline functions undergo error checking while compilation while on the other hand since macros are not really functions they do not undergo usual error checking during compilation. So if we are not careful while writing macros results can be incorrect and go unchecked.

1.8 Given the following code fragment:

```
int main()
{
    float abc(float, float);
                …
                …
}
void qaz(void)
{
     float x, y, s;
     cin >>x;
     cin >> y;
     ….
     s = abc(x,y);
}
```

Will the above function work? Why?

SOL. No the function abc() will not work because it is not a global function
The function abc() is declared inside main() function so it is not accessible from outside main and that is why it will not work.

1.9 When are the default arguments required in a function?

SOL. A default argument is a default value provided by the function parameter.
Default arguments are required in a function when the user does not supply an explicit argument for a parameter. If the user does supply an argument for the parameter ,the user-supplied argument is used.

1.10 Can the effect of default arguments be achieved by overloading functions? How?

SOL. Yes, the effect of default arguments can be achieved by overloading functions.

Let's take an example of returning a sum of two or three numbers.

Using default arguments:

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

int sum (int,int,int);

int main()
{
int a=3,b=4,c=5;
// for sum of two numbers
sum(a,b);

// for sum of three numbers
sum(a,b,c);
return 0;
}

int sum(int h=0, int I=0, int j=0)
{       return (h+i+j);      }
```

Using overloading function:

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

int sum(int,int);
int sum(int,int,int);

int main()
{
int a=3,b=4,c=5,d=6;
//for sum of two
sum(a,b);

//for sum three
sum(a,b,c);
return 0;
}

int sum(int x, int y)
{       return (x+y);                   }

int sum(int x, int y, int z)
{       return (x+y+z);     }
```

1.11 Write a declaration for a function called funct() that takes two arguments and returns a char value. The first argument is int and cannot be modified. The second argument is float with a default value of 3.14.

SOL. char funct(const int, float=3.14);

1.12 Create two identical functions, f1( ) and f2( ).Inline f1( ) and leave f2( ) as an non-inline function. Use the Standard C++ Library function to mark the starting point and ending points and compare the two functions to see which one is faster.

SOL. inline char f1(const int y , float x = 3.14)

```
{
        return y*x;
}
char f2(const int y, float x =3.14)
{
        return y*x;
}
```

1.13 Identify the problem with the following code:

```
void large(int &i, int &j);
int main()
{
    large (1,2);
}

void large(int &a, int &b)
{
    if (a > b)
    a = -1;
    else b= -1;
}
```

How can we correct it?

SOL. Problem : Constants are passed as arguments in function call large(1,2); however reference variables shall be passed.

Solution : function call statement should be written as:

```
…
{
        int a=1,b=2;
        large(a,b);
        return 0;
}
…
```

1.14 Write a function that computes xy by using recursion. Write a C++ main() function to test it. (Hint: 54 = 5 * 53).

SOL.

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

int power(int,int);
int power(int a,int b)
{
  if(b==0)
  { return 1; }
  else
  {
    return a*power(a,b-1);
  }
}
int main()
{
  cout<<power(2,5)<<endl;
  return 0;
}
```

1.15 What is overloading of a function? When can it be useful?

SOL. Function overloading allows multiple functions that provide a common operation on different parameter types to share a common name. Function overloading is one of the most powerful features of C++ programming language and forms the basis of compile time polymorphism. It is used for code reusability and also to save memory.

1.16 GCD function can be implemented in nonrecursive way using iterations. Write the function which calculates GCD in iterative way.

SOL.

```
#include <iostream>

using namespace std;
Int hcf(int a,int b)
{
	While(a != b)
	{
		If(a>b)

	}
}
```