LAB 3

i. WAP to find area of a circle, a rectangle and a triangle, using concept of function overloading

```
#include <iostream>
#include <math.h>
#define pi 3.14

using namespace std;

void area(int r);
void area(int l, int b);
void area(int a, int b, int c);

int main()
{
    int a_167, b_167, c_167;
    cout << "Enter Radius of Circle: ";
    cin >> a_167;
    area(a_167);
    cout << "Enter Length Breadth of Rectangle: ";
    cin >> a_167 >> b_167;
    area(a_167, b_167);
    cout << "Enter sides of triangle: ";
    cin >> a_167 >> b_167 >> c_167;
    area(a_167, b_167, c_167);
    return 0;
}

void area(int r)
{
    cout << "Area of Circle= " << (pi * r * r) << endl;
}

void area(int l, int b)
{
    cout << "Area of Rectangle= " << (l * b) << endl;
}

void area(int a, int b, int c)
{
    //sum of two sides must be greater or eaqual than third
    if ((a + b <= c) || a + c <= b || b + c <= a)
    {
        printf("Not a valid triangle");
        exit(0);
    }
    float s = (a + b + c) / 2;
```

```
    cout << "Area of Triangle= " << sqrt(s * (s - a) * (s - b) * (s - c
)) << endl;
}

Output:

Enter Radius of Circle: 2
Area of Circle= 12.56
Enter Length Breadth of Rectangle: 2 4
Area of Rectangle= 8
Enter sides of triangle: 3 4 5
Area of Triangle= 6
PS T:\KIIT\OOP\OOP LAB>
```

ii. WAP to find volume of a sphere, a cylinder and a cuboid, using function overloading.

```
#include <iostream>
#include <math.h>
#define pi 3.14

using namespace std;

void volume(int r);
void volume(int l, int b);
void volume(int a, int b, int c);

int main()
{
    int a_167, b_167, c_167;
    cout << "Enter Radius of sphere: ";
    cin >> a_167;
    volume(a_167);
    cout << "Enter Length radius of cylinder: ";
    cin >> a_167 >> b_167;
    volume(a_167, b_167);
    cout << "Enter length breadth height of cuboid: ";
    cin >> a_167 >> b_167 >> c_167;
    volume(a_167, b_167, c_167);
    return 0;
}

void volume(int r)
{
```

```
    cout << "Volume Of Sphere= " << (4 * pi * r * r * r) / 3 << endl;
}

void volume(int l, int r)
{
    cout << "Volume Of Cylinder= " << (pi * r * r * l) << endl;
}

void volume(int a, int b, int c)
{
    cout << "Volume of Cuboid= " << (a * b * c) << endl;
}


OUTPUT:

Enter Radius of sphere: 3
Volume Of Sphere= 113.04
Enter Length radius of cylinder: 5 2
Volume Of Cylinder= 62.8
Enter length breadth height of cuboid: 3 4 5
Volume of Cuboid= 60
PS T:\KIIT\OOP\OOP LAB>
```

iii. WAP which displays a given character, n number of times, using a function. When the n value is not provided, it should print the given character 80 times. When both the character and n value is not provided, it should print '*' character 80 times.
[Write the above program in two ways:-
-using function overloading.
-using default arguments.]

```
//default argument
#include <iostream>

using namespace std;

void output(char c_167 = '*', int n_167 = 80)
{
    for (int i = 0; i < n_167; i++)
    {
        cout << c_167;
    }
}

int main()
```

```
{
    int a_167 = 10;
    char b_167 = '%';
    output(b_167, a_167);
    cout << endl;
    output();
    cout << endl;
    output('#');
    return 0;
}

OUTPUT:

%%%%%%%%%
**************************************************************************
##########################################################################
PS T:\KIIT\OOP\OOP LAB>
```

```
//function overloading
#include <iostream>

using namespace std;

void output(char c_167, int n_167)
{
    for (int i = 0; i < n_167; i++)
    {
        cout << c_167;
    }
}

void output(char c_167)
{
    for (int i = 0; i < 80; i++)
    {
        cout << c_167;
    }
}

void output(int n_167)
{
    for (int i = 0; i < n_167; i++)
    {
        cout << "*";
    }
}
```

```
void output()
{
    for (int i = 0; i < 80; i++)
    {
        cout << "*";
    }
}

int main()
{
    int a_167 = 10;
    char b_167 = '%';
    output(b_167, a_167);
    cout << endl;
    output();
    cout << endl;
    output('#');
    return 0;
}


OUTPUT:

%%%%%%%%%%
********************************************************************************
################################################################################
PS T:\KIIT\OOP\OOP LAB>
```

iv. WAP to find square and cube of a number using inline function.
Inline int sq(int I) { return I*I;}

```
#include <iostream>

using namespace std;

inline int square(int n)
```

```
{
    return n * n;
}

inline int cube(int n)
{
    return n * n * n;
}

int main()
{
    cout << "Enter Number: ";
    int n_167;
    cin >> n_167;
    cout << "Square = " << square(n_167) << endl;
    cout << "Cube = " << cube(n_167) << endl;
    return 0;
}

OUTPUT:

Enter Number: 4
Square = 16
Cube = 64
PS T:\KIIT\OOP\OOP LAB>
```

v. WAP to increment the value of an argument given to function USING INLINE function.

Inline int incr(int I) { return ++i; }

```
#include <iostream>

using namespace std;

inline int increment(int n)
{
    return ++n;
}

int main()
{
    cout << "Enter Number: ";
    int n_167;
    cin >> n_167;
    cout << "Incremented value = " << increment(n_167) << endl;
    return 0;
}
```

```
OUTPUT:

Enter Number: 60
Incremented value = 61
PS T:\KIIT\OOP\OOP LAB>
```

Vi. Write a program to create a class called COMPLEX and implement the following overloading functions ADD that return a COMPLEX number.

a) ADD (int a , complex s2) - where a is an integer (real part) and s2 is a complex number. b) ADD (s1, s2) - where s1 and s2 are complex numbers.

```cpp
#include <iostream>

using namespace std;

class complex
{
    int real;
    int img;

public:
    void getdata()
    {
        cout << "Enter real part: ";
        cin >> real;
        cout << "Enter Imaginnary part: ";
        cin >> img;
    }
    void display()
    {
        cout << real << " +i" << img << endl;
    }
    void ADD(int a, complex s)
    {
        real = s.real + a;
        img = s.img;
    }
    void ADD(complex s1, complex s2)
    {
        real = s1.real + s2.real;
        img = s1.img + s2.img;
    }
};
```

```
int main()
{
    complex a_167, b_167, c_167;
    a_167.getdata();
    int r_167;
    cout << "Enter a real no: ";
    cin >> r_167;
    b_167.ADD(r_167, a_167);
    c_167.ADD(a_167, b_167);
    b_167.display();
    c_167.display();
    return 0;
}

OUTPUT:

Enter real part: 2
Enter Imaginnary part: 3
Enter a real no: 1
3 +i3
5 +i6
PS T:\KIIT\OOP\OOP LAB>
```

vii. Write a program to find the summation of three numbers by using one function only with function name SUM having three arguments. If at run time one argument is given to the function SUM, then second and third argument will be assumed by default as 10 and 20 respectively. If two arguments are given at run time, then third argument will be assumed as 20. Use function with default argument concepts.

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

void SUM(int a, int b = 10, int c = 20)
{
    cout << a + b + c << endl;
}

int main()
{
    int a_167, b_167, c_167;
    cout << "Enter three numbers: ";
    cin >> a_167 >> b_167 >> c_167;
    cout << "3 Args: ";
    SUM(a_167, b_167, c_167);
    cout << "2 Args: ";
```

```
    SUM(a_167, b_167);
    cout << "1 Args: ";
    SUM(a_167);
    return 0;
}

OUTPUT:

Enter three numbers: 2 3 4
3 Args: 9
2 Args: 25
1 Args: 32
PS T:\KIIT\OOP\OOP LAB>
```

Viii. Write a program to demonstrate the concept of call-by-value, call-by-reference & call-by address by taking swapping of two numbers as an example.

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

void swappingVal(int a, int b)
{ //by-value
    a += b;
    b = a - b;
    a -= b;
    cout << "In func " << a << " " << b << endl;
}

void swappingAdd(int *a, int *b)
{ //by-Address
    *a += *b;
    *b = *a - *b;
    *a -= *b;
}

void swappingReff(int &p, int &q)
{ //by-reff
    p += q;
    q = p - q;
    p -= q;
}

int main()
{
```

```
    int a_167, b_167;
    cout << "Enter two numbers: ";
    cin >> a_167 >> b_167;
    cout << "Call by value " << endl;
    swappingVal(a_167, b_167);
    cout << "In main" << a_167 << " " << b_167 << endl;
    cout << "Call by Address ";
    swappingAdd(&a_167, &b_167);
    cout << a_167 << " " << b_167 << endl;
    cout << "Call by Reff ";
    swappingReff(a_167, b_167);
    cout << a_167 << " " << b_167 << endl;
}

OUTPUT:

Enter two numbers: 4 5
Call by value
In func 5 4
In main4 5
Call by Address 5 4
Call by Reff 4 5
PS T:\KIIT\OOP\OOP LAB>
```

Ix. Write a program to demonstrate the use of scope resolution operator(::) by declaring same variable name globally and locally and display the value of global and local variables.

```
#include <iostream>
using namespace std;
int x = 10; // Global x
int main()
{
    int x = 20; // Local x
    {
        int k = x;
        int x = 30;
        cout << "k = " << k << endl;
        cout << "x = " << x << endl;
        cout
            << "::x = " << ::x << endl;
    }
    cout << "x = " << x << endl;
    cout
        << "::x = " << ::x << endl;
```

```
    return 0;
}

OUTPUT:
k = 20
x = 30
::x = 10
x = 20
::x = 10
PS T:\C++\CPP\3rd SEM\OOP LAB\LAB 3\9>
```