# PPL Assignment 4

Name: Himani Verma

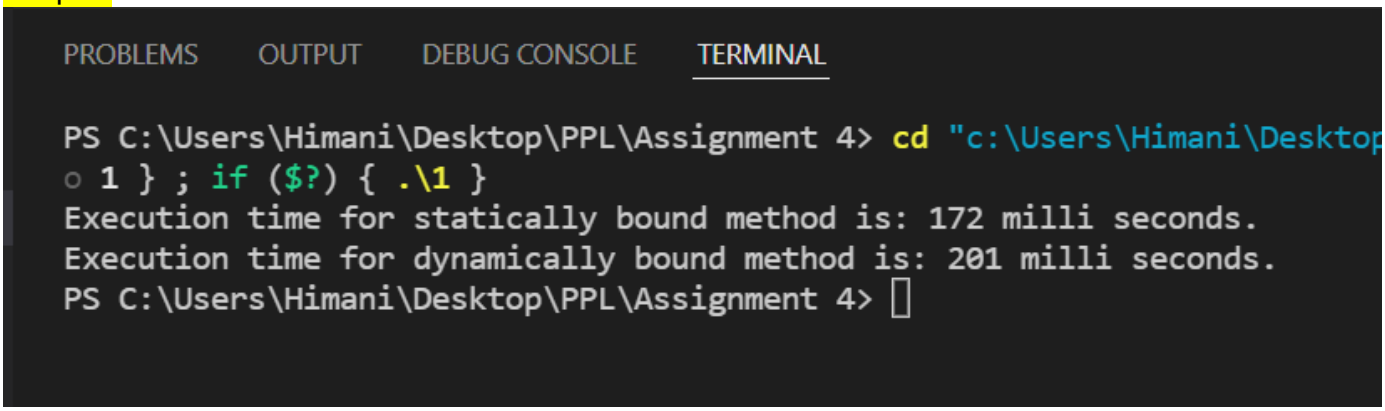
Admission No: U19CS075

1. Write a program in C++ that calls both a dynamically bound method and a statically bound method a large number of times, timing the calls to both of the two. Compare the timing results and compute the difference of the time required by the two. Explain the results.

Source Code:

```
#include<iomanip>
#include<cstring>
#include<cstdlib>
#include<ctime>
#include<iostream>
using namespace std;
const int pi=3.14;
class circle{
    public:
        double perimeter_static(){
            return 2*pi*20;
        }
        virtual double perimeter(){
            return 2*pi*4;
        }
};
class semicircle: public circle{
    public:
        double perimeter(){
            return pi*4;
        }
};
const int calls= 100000000;
int main(){
    circle* c = new semicircle();
    int start_static =clock();
    for(int i=0; i<calls; i++){
        c->perimeter_static();
    }
    int stop_static= clock();
    cout<<"Execution time for statically bound method is: "<<(stop_static-
start_static)/(double(CLOCKS_PER_SEC))*1000<<" milli seconds."<< endl;
    int start_dynamic =clock();
    for(int i=0; i<calls; i++){
        c->perimeter();
    }
    int stop_dynamic= clock();
    cout<<"Execution time for dynamically bound method is: "<<(stop_dynamic-
start_dynamic)/(double(CLOCKS_PER_SEC))*1000<<" milli seconds."<< endl;
    return 0;
}
```

Output:



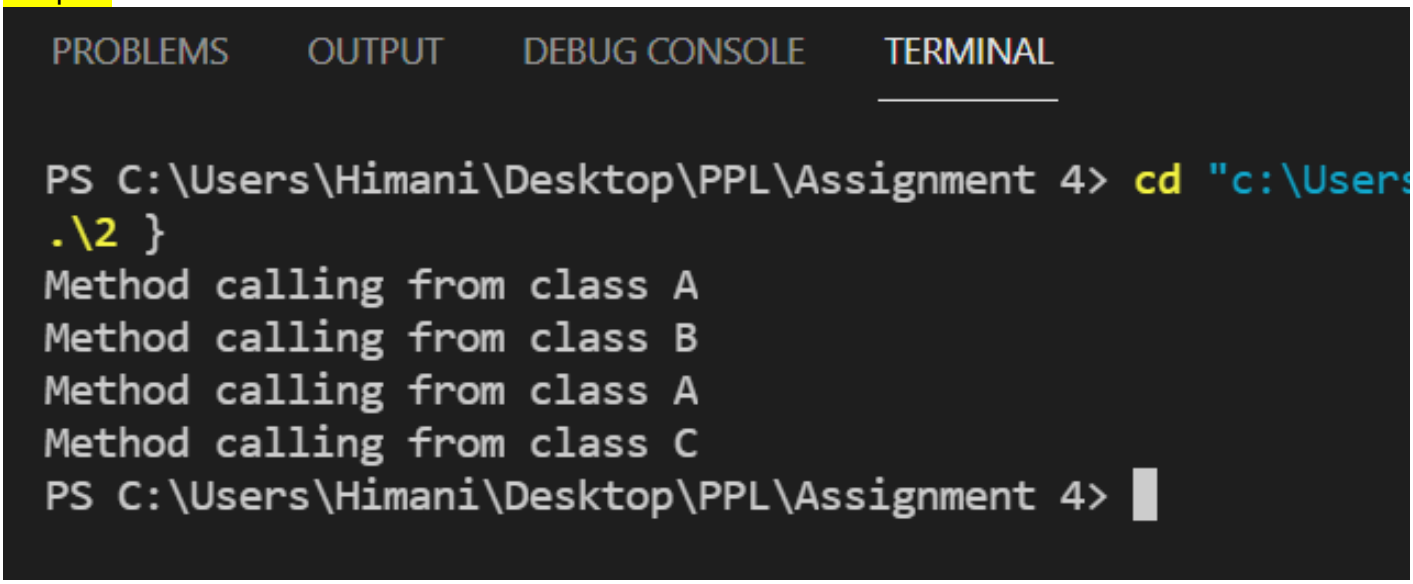

2. Design and implement a C++ program that defines a base class A, which has a subclass B, which itself has a subclass C. The A class must implement a method, which is overridden in both B and C. You must also write a test class that instantiates A, B, and C and includes three calls to the method. One of the calls must be statically bound to A's method. One call must be dynamically bound to B's method, and one must be dynamically bound to C's method. All of the method calls must be through a pointer to class A.

Source Code:

```
#include<iostream>
#include<stdio.h>
using namespace std;
class A{
    public:
            virtual void m(){
                cout<<"Method calling from class A"<<endl;
            }
            A(){
                m();
            }
};
class B: public A{
    public:
            void m(){
                cout<<"Method calling from class B"<<endl;
            }
};
class C: public B{
    public:
            void m(){
                cout<<"Method calling from class C"<<endl;
            }

};
int main(){
    A* a= new B();
    a->m();
    a=new C();
    a->m();
    return 0;
}
```

Output:




3. Consider the following C++ skeletal program:

```
class Big
{
        int i; float f;
        void fun1() throw int {
        . . . try {
        . . . throw i;
        . . . throw f;
        . . . }
        catch(float) {   <------1
         . . . }
        . . . }
        }
class Small {
int j; float g;
        void fun2() throw float
        { . . . try {
         . . . try {
        Big.fun1();
        . . . throw j;
        . . . throw g;
        . . . }
        catch(int) {  <------2
        . . . }
         . . . }
        catch(float) {  <------3
        . . . } } }
```

In each of the four throw statements, where is the exception handled? Note that fun1 is called from fun2 in class Small.

Solution:

**throw i :** This throw statement will be handled by catch(int){...} (2). As the data type of variable i is int, when throw i statement generates error it will be handled by catch 2 as it is nearest matching catch(int) block.

**throw f :** This throw statement will be handled by catch(float){...} (1).

**throw j :** This throw statement will be handled by catch(int){...} (2).

**throw g :** This throw statement will be handled by catch(float){...} (3).

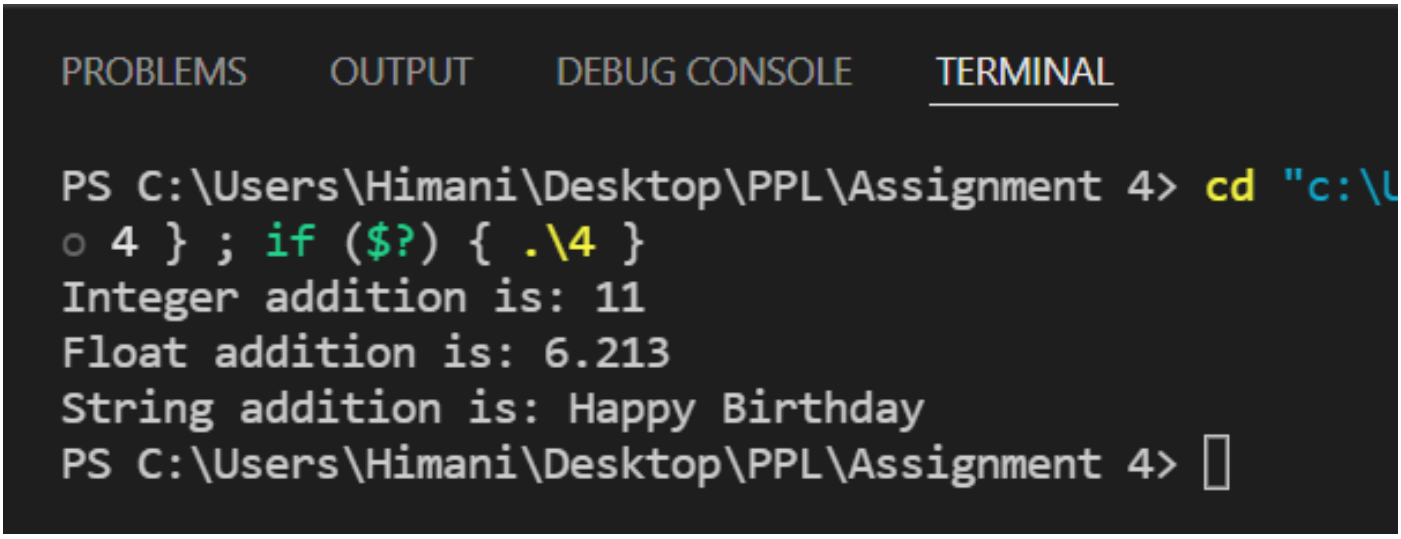
4. Write a C++ program that takes a set of inputs. The type of input governs the kind of operation to be performed, i.e. concatenation for strings and addition for int or float. You need to write the class template AddElements which has a function add() for giving the sum of int or float elements. You also need to write a template specialization for the typed string with a function concatenate() to concatenate the second string to the first string.

Source Code:

```
#include<iostream>
using namespace std;
template <class T>
class AddElements{
    public:
        T element;
        AddElements(T args){
            element=args;
        }
        T add(T ele){
            return (element+ele);
        }
};

// Class Template Specialization
template <>
class AddElements <string>{
    public:
        string element;
        AddElements(string args){
            element = args;
    }
         string concatenate(string ele){
            return (element + ele);
    }
};
int main(){
    AddElements<int> a(4);
    cout<<"Integer addition is: "<<a.add(7)<<endl;
    AddElements<float> b(3.763);
    cout<<"Float addition is: "<<b.add(2.45)<<endl;
    AddElements<string> c("Happy");
    cout<<"String addition is: "<<c.concatenate(" Birthday")<<endl;
    return 0;
}
```

Output:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PS C:\Users\Himani\Desktop\PPL\Assignment 4> cd "c:\U
o 4 } ; if ($?) { .\4 }
Integer addition is: 11
Float addition is: 6.213
String addition is: Happy Birthday
PS C:\Users\Himani\Desktop\PPL\Assignment 4>