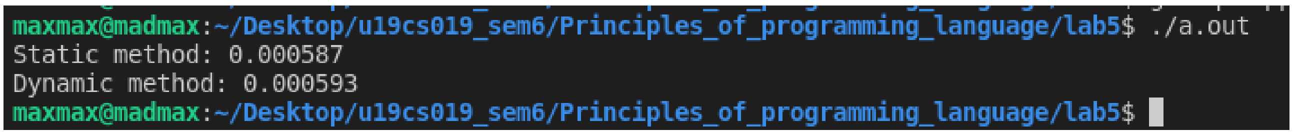
1. Write a program in C++ that calls both a dynamically bound method and a statically bound method a large number of times, timing the calls to both of the two. Compare the timing results and compute the difference of the time required by the two. Explain the results.

```
#include <iostream>
#include <bits/stdc++.h>

using namespace std;

class A {
public:
double static_method(double x) {
return x * x;
}
virtual double dynamic_method(double x) {
return x * x;
}
};

class B: public A {
public:
double dynamic_method(double x) {
return x * x;
}
};

const int N = 100000;

int main () {
A *a = new B();
time_t start, end;
start = clock();
for (int i = 0; i < N; i++) {
a->static_method(i);
}
end = clock();
cout << "Static method: " << (double)(end - start) / CLOCKS_PER_SEC << endl;
start = clock();
for (int i = 0; i < N; i++) {
a->dynamic_method(i);
}
end = clock();
cout << "Dynamic method: " << (double)(end - start) / CLOCKS_PER_SEC << endl;
return 0;
}
```

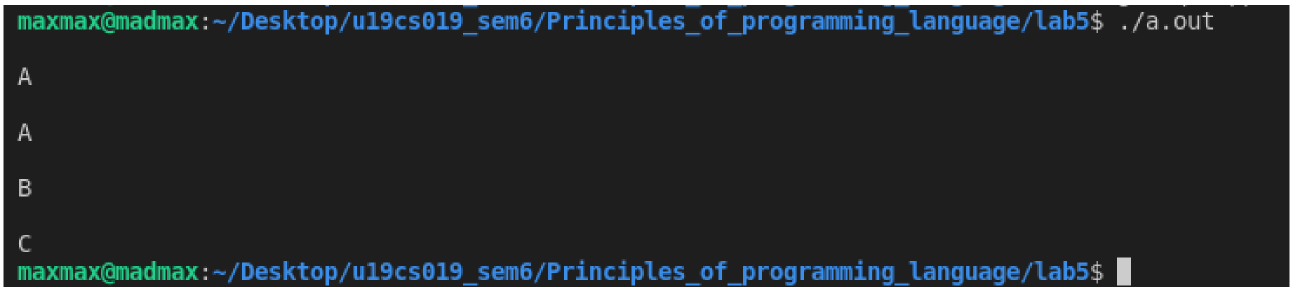


```
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab5$ ./a.out
Static method: 0.000587
Dynamic method: 0.000593
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab5$
```

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.

```
#include <iostream>

using namespace std;

class A
{
public:
virtual void print()
{
cout << "\nA\n";
}
};

class B : public A
{
public:
void print()
{
cout << "\nB\n";
}
};

class C : public B
{
public:
void print()
{
cout << "\nC\n";
}
};

int main()
```

```
{
A a, *test;
B b;
C c;
a.print(); // statically bound to A
test = &a;
test->print();
test = &b;
test->print(); // dynamically bound to B
test = &c;
test->print(); // dynamically bound to C
return 0;
}
```



```
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab5$ ./a.out

A

A

B

C
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab5$
```

3. Consider the following C++ skeletal program:

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

```
. . . throw j;
. . . throw g;
. . . }
catch(int) {
. . . }
 . . . }
catch(float) {
. . . } } }
```

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

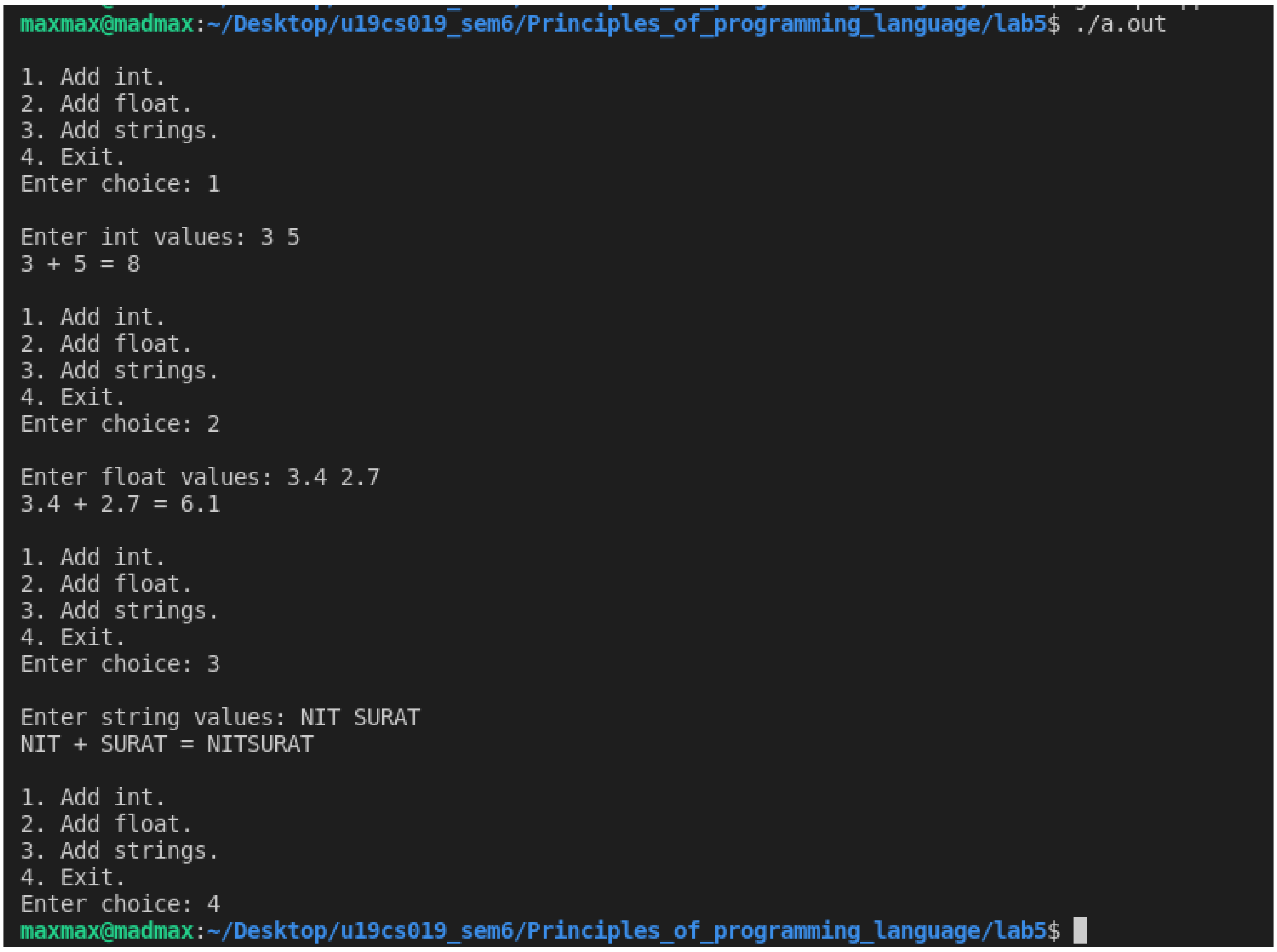
‘I’ IS CAUGHT BY CATCH 2
‘F’ IS CAUGHT BY CATCH 1
‘J’ IS CAUGHT BY CATCH 2
‘G’ IS CAUGHT BY CATCH 3

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.

```
#include <iostream>

using namespace std;

template <class T1, class T2>

class AddElements{
T1 a;
T2 b;
public:
AddElements(T1 a1, T2 b1)
{
a = a1;
b = b1;
};
void add()
{
cout << a << " + " << b << " = " << a + b << endl;
}
};
```

```
template <> // class template specialization

class AddElements<string, string> {
string a, b;
public:
AddElements(string a1, string b1)
{
a = a1;
b = b1;
}
void concatenate()
{
cout << a << " + " << b << " = " << a + b << endl;
}
};

int main(){
int ch;
while (1)
{
cout << "\n1. Add int.";
cout << "\n2. Add float.";
cout << "\n3. Add strings.";
cout << "\n4. Exit.";
cout << "\nEnter choice: ";
cin >> ch;
if (ch == 1)
{
int a, b;
cout << "\nEnter int values: ";
cin >> a >> b;
AddElements<int, int> p(a, b);
p.add();
}
else if (ch == 2)
{
float a, b;
cout << "\nEnter float values: ";
cin >> a >> b;
AddElements<float, float> q(a, b);
q.add();
}
else if (ch == 3)
{
```

```
string c, d;
cout << "\nEnter string values: ";
cin >> c >> d;
AddElements<string, string> s(c, d);
s.concatenate();
}
else
exit(0);
}
return 0;
}
```



```
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab5$ ./a.out

1. Add int.
2. Add float.
3. Add strings.
4. Exit.
Enter choice: 1

Enter int values: 3 5
3 + 5 = 8

1. Add int.
2. Add float.
3. Add strings.
4. Exit.
Enter choice: 2

Enter float values: 3.4 2.7
3.4 + 2.7 = 6.1

1. Add int.
2. Add float.
3. Add strings.
4. Exit.
Enter choice: 3

Enter string values: NIT SURAT
NIT + SURAT = NITSURAT

1. Add int.
2. Add float.
3. Add strings.
4. Exit.
Enter choice: 4
maxmax@madmax:~/Desktop/u19cs019_sem6/Principles_of_programming_language/lab5$
```