LAB 15

1. WAP to show following

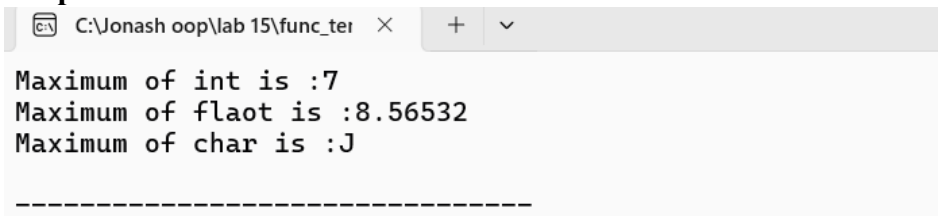
i) Function template to find the maximum value of generic type.

ii) Function template to find minimum vale with multiple arguments.

i)

```
#include <iostream>
using namespace std;
template <typename A>
A maximum(A n1, A n2)
{
   if (n1 > n2)
      return n1;
   else
      return n2;
}
int main ()
{
   int n1 = 1, n2 = 7;
   float f1 = 3.9087, f2 = 8.56532;
   char c1 = 'J', c2 = 'A';
   cout << "Maximum of int is :" << maximum(n1, n2) << endl;
   cout << "Maximum of flaot is :" << maximum(f1, f2) << endl;
   cout << "Maximum of char is :" << maximum(c1, c2) << endl;
}
```

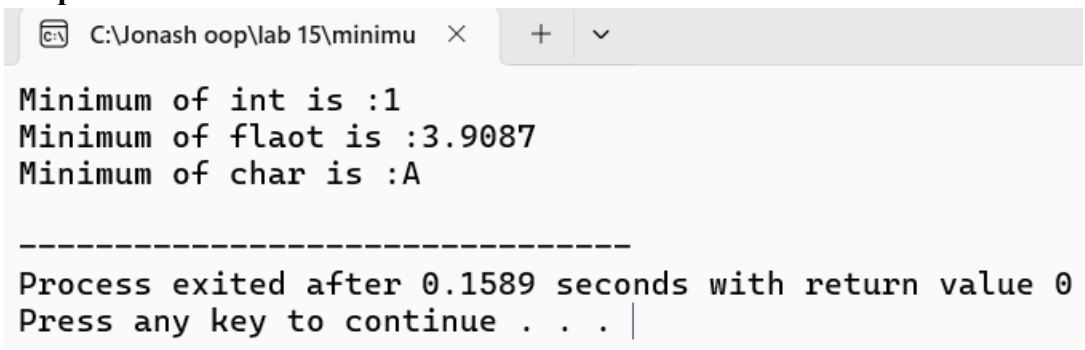
Output:



ii)

```
#include <iostream>
using namespace std;
template <typename A, typename B>
A minimum(A n1, B n2)
{
   if (n1 > n2)
      return n2;
   else
      return n1;
}
int main()
{
   int n1 = 1, n2 = 7;
   float f1 = 3.9087, f2 = 8.56532;
   char c1 = 'J', c2 = 'A';
   cout << "Minimum of int is :" << minimum(n1, n2) << endl;
   cout << "Minimum of flaot is :" << minimum(f1, f2) << endl;
   cout << "Minimum of char is :" << minimum(c1, c2) << endl;
}
```

Output:

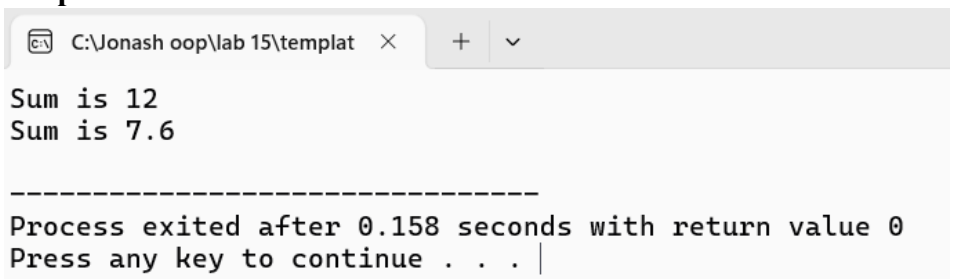


2. WAP to show template class.

```
#include <iostream>
using namespace std;
template <class T>
class demo
{
   T a, b, c;

public:
   demo(T x, T y, T z)
   {
      a = x;
      b = y;
      c = z;
   }
   void sum()
   {
      cout << "Sum is " << a + b + c << endl;
   }
};
int main()
{
   demo<int> aa(1, 7, 4);
   aa.sum();
   demo<float> bb(1.2, 2.8, 3.6);
   bb.sum();
   return 0;
}
```

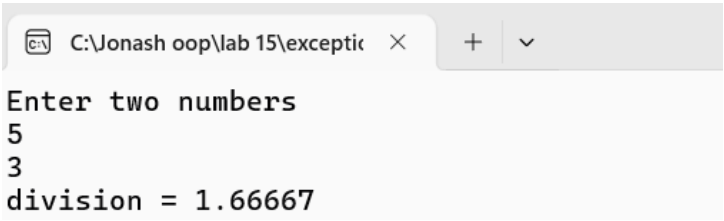
Output:



3. WAP containing possible exception. Use try block to throw it and a catch block to handle it promptly.

```
#include <iostream>
using namespace std;
int main()
{
   int a, b;
   cout << "Enter two numbers " << endl;
   cin >> a >> b;
   try
   {
      if (b == 0)
         throw b;
      else
         cout << "division = " << (float)a / b << endl;
   }
   catch (int e)
   {
      cout << "Error zero cannot be divided!!!" << endl;
   }
}
```

Output:

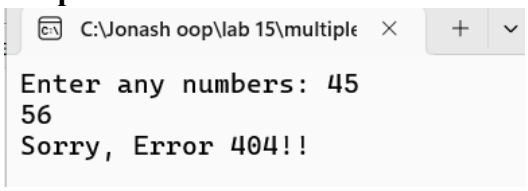


4. WAP to illustrate multiple catch and catch(...) [ellipse] handle

```
#include <iostream>
using namespace std;
int main()
{
   int n1, n2;
   cout << "Enter any numbers: ";
   cin >> n1 >> n2;
   try
   {
      if (n1 > 100)
         cout << n1 << " is greater than 100" << endl;
      else
         throw 111;
   }
   catch (char z)
   {
      cout << "No exception " << z << endl;
   }
   catch (...)
   {
      cout << "Sorry, Error 404!!" << endl;
```

```
    }
}
```

Output:



5. WAP to illustrate exception handling in function.

```
#include <iostream>
using namespace std;
void excep(int x, int y)
{
   try
   {
      if (x > 100)
         cout << x << " is greater than 100" << endl;
      else
         throw 111;
   }
   catch (char z)
   {
      cout << "No exception" << z << endl;
   }
   catch (...)
   {
      cout << "Sorry, error 404!!" << endl;
   }
}
int main()
{
   int n1, n2;
   cout << "Enter the values" << endl;
   ;
   cin >> n1 >> n2;
   excep(n1, n2);
}
```

Output: