**C++ Assignment [17-01-2018]**

|  |  |
| --- | --- |
| **Emp Name** | **Program And Output** |
| **1\_Anan\_Mishra** | **/\* Exception Handling: try and Catch with integer \*/**    #include <iostream>  using namespace std;  int main()  {  int x = -1;  // Some code  cout << "Before try \n";  try {  cout << "Inside try \n";  if (x < 0)  {  throw x;  cout << "After throw (Never executed) \n";  }  }  catch (int x ) {  cout << "Exception Caught \n";  }  cout << "After catch (Will be executed) \n";  return 0;  }  -----------------------------------------------------------------------------------------  **Output:**  **Before try**  **Inside try**  **Exception Caught**  **After catch (Will be executed)** |
| **2\_Harish** | **/\* Exception Handling: try and catch using num and string \*/** |
| **3\_Himabindu** | **/\* Exception Handling: try and Catch with character \*/**  #include<iostream>  #include <string>  using namespace std;  int main ()  {   int num;  string str\_bad = "wrong number used";  cout << "Input 1 or 2: ";  cin >> num;  try  {  if ( num == 1 )  {  throw 5;  }  if ( num == 2 )  {  throw str\_bad;  }  }  catch (int a)  {  cout << "An exception occurred!" << endl;  cout << "Exception number is: " << a << endl;  }  catch (string b)  {  cout << "An exception occurred!" << endl;  cout << "Exception number is: " << b << endl;  }  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output 1:**  **Input 1 or 2: 1**  **An exception occurred!**  **Exception number is: 5**  **Output 2:**  **Input 1 or 2: 2**  **An exception occurred!**  **Exception number is: wrong number used** |
| **4\_Divya** | **/\*Exception Handling: Nested try and catch \*/**  #include <iostream>  using namespace std;    int main()  {  try {  try {  throw 20;  }  catch (int n) {  cout << "Handle Partially ";  throw; //Re-throwing an exception  }  }  catch (int n) {  cout<< n <<endl;  cout << "Handle remaining ";  }  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **Handle Partially 20**  **Handle remaining** |
| **5\_Deepika** | **/\* Excption Handling: Try and Catch with Constructor and Destructor \*/**  #include <iostream>  using namespace std;  class Test1 {  public:  Test1()  {  cout << "Constructing an Object of Test1" << endl;  }  ~Test1()  {  cout << "Destructing an Object of Test1" << endl;  }  };  class Test2 {  public:  Test2()  {  cout << "Constructing an Object of Test2" << endl;  throw 20;  }  ~Test2()  {  cout << "Destructing an Object of Test2" << endl;  }  };  int main()  {  try {  Test1 t1;  Test2 t2;  Test1 t3;  }  catch(int num)  {  cout << "Caught " << num << endl;  }  return 0;  **}**  **-----------------------------------------------------------------------------------------**  **Output:**  **Constructing an Object of Test1**  **Constructing an Object of Test2**  **Destructing an Object of Test1**  **Caught 20** |
| **6\_Ramya** | **/\* Excption Handling: Try and Catch with inheritance \*/**  #include<iostream>  using namespace std;  class Base  {  };  class Derived: public Base  {  };  int main()  {  Base b;  Derived d;  try  {  throw d;  }  catch(Derived d) {  cout<<"Caught Derived Exception"<<endl;  }  catch(Base b)  {  cout<<"Caught Base Exception";  }  return 0;  **}**  **-----------------------------------------------------------------------------------------**  **Output:**  **Caught Derived Exception** |
| **7\_Ramya** | **/\* Excption Handling: Try and Catch with inheritance in reverse order\*/**  #include<iostream>  using namespace std;  class Base  {  };  class Derived: public Base  {  };  int main()  {  Base b;  Derived d;  try  {  throw d;  }  catch(Base b)  {  cout<<"Caught Base Exception"<<endl;  }  catch(Derived d) {  cout<<"Caught Derived Exception"<<endl;  }  return 0;  }  **----------------------------------------------------------------------------------------**  **Output:**  **Caught Base Exception** |
| **8\_Rahul** | **/\* Templet function \* /**  #include<iostream>  using namespace std;  template <class myvar>  myvar Getmax(myvar val1, myvar val2)  {  myvar res;  res=(val1 > val2 ) ? val1 : val2;  return res;  }  int main()  {  int var1=10, var2=20,res;  long num1=400, num2=300,res1;  char ch1='A',ch2='D',res2;  res=Getmax(var1,var2);  res1=Getmax(num1,num2);  res2=Getmax(ch1,ch2);  cout << "Result of Int Variable : "<<res <<endl;  cout << "Result of long Variable : "<<res1 <<endl;  cout << "Result of char Variable : "<<res2 <<endl;  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **Result of Int Variable : 20**  **Result of long Variable : 400**  **Result of char Variable : D** |
| **9\_Swetha** | **/\* Class Templet \*/**  #include<iostream>  using namespace std;  template <class T>  class Calculator  {  private:  T num1,num2;  public:  Calculator(T n1, T n2)  {  num1=n1;  num2=n2;  }  void display()  {  cout<<"numbers are:"<< num1<<" and "<< num2<<endl;  cout<<"Addition is :"<<add() <<endl;  cout<<"substraction is:" <<substract()<<endl;  cout<<"Product is :"<< multiply()<<endl;  cout<<"Division is :"<<divide()<<endl;  }  T add()  {  return num1+num2;  }  T substract()  {  return num1-num2;  }  T multiply()  {  return num1\*num2;  }  T divide()  {  return num1/num2;  }  };  int main()  {  Calculator<int> intCal(2,2);  Calculator<float> floatCal(2.4,1.1);  cout<<"Integer results:"<<endl;  intCal.display();  cout<<endl<<"Float results:"<<endl;  floatCal.display();  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **Integer results:**  **numbers are:2 and 2**  **Addition is :4**  **substraction is:0**  **Product is :4**  **Division is :1**  **Float results:**  **numbers are:2.4 and 1.1**  **Addition is :3.5**  **substraction is:1.3**  **Product is :2.64**  **Division is :2.18182** |