**WEEK-9**

**NAME – SANCHIT JAIN**

**BATCH – B-7**

**ENROLL – 21103192**

#include <iostream>

using namespace std;

int main()

{

    int n1;

    cout << "Enter an integervalue : "; cin >> n1; char c1;

    cout<< "Enter a character value : ";

    cin >> c1;

    double d1;

    cout << "Enter a double value : ";

    cin >> d1;

    try

    {

        if (n1 == 10)

        {

            throw n1;

        }

        else

        {

            cout << "YOUR VALUE IS:" << n1 << endl;

        }

    }

    catch (int x)

    {

        cout << "YOU CAN'T ENTER 10" << endl;

    }

    try

    {

        if (c1 == 'z')

        {

            throw c1;

        }

        else

        {

            cout << "YOUR CHARACTER IS:" << c1 << endl;

        }

    }

    catch (char c)

    {

        cout << "YOU CAN'T ENTER z" << endl;

    }

    try

    {

        if (d1 == 2.1)

        {

            throw d1;

        }

        else

        {

            cout << "YOUR VALUE IS:" << d1 << endl;

        }

    }

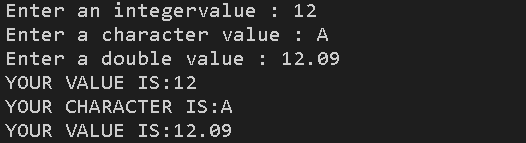
    catch (double d)

    {

        cout << "YOU CAN'T ENTER 2.1" << endl;

    }

}

****

**Q2). Output** –

Try block entered.

Exception thrown with

wait\_time equal to 46

After catch block.

**Q3). Output –**

Try block entered.

Leaving try block.

After catch block.

**Q4) Ans** – In C++ throw; when executed inside a catch block rethrows the currently caught exception outside the block.

**Q5). Output –**

Trying. Starting sample\_function.

Catching. End of program.

**Q6). Output** –

Trying. Starting sample\_function.

Trying after call. End of program.

**Q7). What happens when an exception is never caught?**

Ans – If an exception is not caught (with a catch block), the runtime system will abort the program (i.e. crash) and an exception message will print to the console.

**Q8). Can you nest a try block inside another try block?**

Ans – Yes, we can declare a try-catch block within another try-catch block, this is called nested try-catch block.

**Q9). Output – (d)**

**Q10). Output – (a**