OOP-Test

With Justifications Write the Output of each of the following programs

1.

```
#include <iostream>
using namespace std;

class Test {
public:
       Test() { cout << "Constructor of Test " << endl; }
       ~Test() { cout << "Destructor of Test "  << endl; }
};

int main() {
    try {
            Test t1;
             throw 10;
      }
      catch(int i)
      {
       cout << "Caught " << i << endl;
      }

}
```

2.

```
#include <iostream>
using namespace std;

int main()
{
       try {
              try{
                     throw 20;
              }
              catch (int n) {
                     cout << "Handle Partially ";
                     throw;
             }
          }
       catch (int n) {
       cout << "Handle remaining ";
```

```
	}
	return 0;
}
```

3.

```
#include <iostream>
using namespace std;

int main()
{
	try {
	throw 'a';
	}
	catch (int x) {
	cout << "Caught ";
	}
	return 0;
}
```

4.

```
#include <iostream>
using namespace std;

int main()
{
	try {
		throw 'a';
		}
	catch (int x) {
	cout << "Caught " << x;
	}
	catch (...) {
	cout << "Default Exception\n";
	}
	return 0;
}
```

5.

```
#include <iostream>
using namespace std;

int main()
{
	try {
		throw 10;
}
	catch (char *excp) {
	cout << "Caught " << excp;
	}
catch (...) {
cout << "Default Exception\n";
}
return 0;
}
```

6.

```
#include <iostream>
using namespace std;

int main()
{
int x = -1;

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;
}
```

7.

```
#include <iostream>
using namespace std;
```

```
void fun(int *ptr, int x)
{
	if (ptr == NULL)
	throw ptr;
	if (x == 0)
	throw x;
	/* Some functionality */
}

int main()
{
	try {
		fun(NULL, 0);
	}
	catch(...) {
	cout << "Caught exception from fun()";
	}
	return 0;
}
```

8.

```
#include <iostream>

using namespace std;
void MyFunc( void );

class CTest
{
public:
    CTest(){};
    ~CTest(){};
    const char *ShowReason() { return "Exception in CTest class."; }

};

class CDtorDemo
{
public:
    CDtorDemo();
    ~CDtorDemo();
};

CDtorDemo::CDtorDemo()
```

```
{
    cout << "Constructing CDtorDemo." << endl;
}

CDtorDemo::~CDtorDemo()
{
    cout << "Destructing CDtorDemo." << endl;
}

void MyFunc()
{

    CDtorDemo D;
    cout<< "In MyFunc(). Throwing CTest exception." << endl;
    throw CTest();
}

int main()
{
    cout << "In main." << endl;
    try
    {
        cout << "In try block, calling MyFunc()." << endl;
        MyFunc();
    }
    catch( CTest E )
    {
            cout << "In catch handler." << endl;
            cout << "Caught CTest exception type: ";
             cout << E.ShowReason() << endl;
    }
    catch( char *str )
    {
        cout << "Caught some other exception: " << str << endl;
    }
    cout << "Back in main. Execution resumes here." << endl;
    return 0;
}
```