

Date:

Roll No.:

2	4	B	1	1	A	1	2	3	3
---	---	---	---	---	---	---	---	---	---

Program No: 11(a)

Develop a C++ program that demonstrates exception handling using try, throw, and catch blocks.

Aim: Develop a C++ program that demonstrates exception handling using try, throw, and catch blocks.

Description:

This program illustrates how exceptions are handled in C++ using try, throw, and catch. When an exceptional condition occurs, the program throws an exception, which is then caught and handled to prevent abnormal termination.

Syntax:

```
try {  
    // code that may throw  
}  
  
catch (exceptionType e) {  
    // handle exception  
}
```



Program:

```
#include <iostream>  
  
using namespace std;  
  
int divide(int a, int b) {  
    if (b == 0) {  
        throw "Division by zero error!";  
    }  
    return a / b;  
}  
  
int main() {
```

A D I T Y A
U N I V E R S I T Y

Date:

Roll No.:

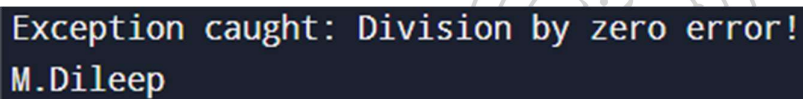
2	4	B	1	1	A	1	2	3	3
---	---	---	---	---	---	---	---	---	---

```
int x = 10, y = 0;

try {
    int result = divide(x, y);
    cout << "Result: " << result << endl;
} catch (const char* msg) {
    cout << "Exception caught: " << msg << endl;
}

cout << "M.Dileep" << endl;
return 0;
}
```

Output:



```
Exception caught: Division by zero error!
M.Dileep
```

```
=== Code Execution Successful ===
```

CO Mapped: CO4

POs Mapped: PO1 , PO2 , PO3 , PO4 , PO5 , PO9 , PO11

PSOs Mapped: PSO1

Date:

Roll No.:

2	4	B	1	1	A	I	2	3	3
---	---	---	---	---	---	---	---	---	---

Program No: 11(b)

Develop a C++ program to illustrate the use of multiple catch statements, where different types of exceptions are caught and handled differently.

Aim: Develop a C++ program to illustrate the use of multiple catch statements, where different types of exceptions are caught and handled differently.

Description:

This program demonstrates multiple catch blocks to handle various exception types. Each catch block can handle a different exception, allowing fine-grained control over error handling.

Syntax:

```
try {  
    // code  
}  
catch(Type1 e) { }  
catch(Type2 e) { }
```

Program:

```
#include <iostream>  
using namespace std;  
  
int main() {  
    try {  
        int choice;  
        cout << "Enter 1 for int exception, 2 for double exception: ";  
        cin >> choice;  
  
        if (choice == 1) throw 100;  
        else if (choice == 2) throw 3.14;  
        else throw "Unknown exception";  
    }  
}
```



A D I T Y A
U N I V E R S I T Y

Date:

Roll No.:

2	4	B	1	1	A	1	2	3	3
---	---	---	---	---	---	---	---	---	---

```
catch (int i) {  
    cout << "Caught an integer exception: " << i << endl;  
}  
catch (double d) {  
    cout << "Caught a double exception: " << d << endl;  
}  
catch (const char* msg) {  
    cout << "Caught a string exception: " << msg << endl;  
}  
  
cout << "M.Dileep" << endl;  
return 0;  
}
```



Output :

```
Enter 1 for int exception, 2 for double exception: 2  
Caught a double exception: 3.14  
M.Dileep  
  
=== Code Execution Successful ===
```

CO Mapped: CO4

POs Mapped: PO1 , PO2 , PO3 , PO4 , PO5 , PO9 , PO11

PSOs Mapped: PSO1

Date:

Roll No.:

2	4	B	1	1	A	I	2	3	3
---	---	---	---	---	---	---	---	---	---

Program No: 12(a)

Develop a C++ program to implement List and Vector containers and perform basic operations such as insertion, deletion, traversal.

Aim: Develop a C++ program to implement List and Vector containers and perform basic operations such as insertion, deletion, traversal.

Description:

This program demonstrates the use of STL containers list and vector, performing operations like inserting elements, deleting elements, and traversing through the container.

Syntax:

```
#include <vector>
```

```
vector<type> v;    // declare vector
```

```
v.push_back(value); // insert at end
```

```
v.pop_back();      // remove last element
```

```
v[i];              // access element at index i
```

```
#include <list>
```

```
list<type> l;      // declare list
```

```
l.push_back(value); // insert at end
```

```
l.push_front(value); // insert at front
```

```
l.pop_back();       // delete last
```

```
l.pop_front();      // delete first
```

Program:

```
#include <iostream>
```

```
#include <vector>
```

```
#include <list>
```

```
using namespace std;
```

Date:

Roll No.:

2	4	B	1	1	A	I	2	3	3
---	---	---	---	---	---	---	---	---	---

```
int main() {  
    // Vector operations  
    vector<int> v = {10, 20, 30};  
    v.push_back(40); // insert  
    cout << "Vector elements: ";  
    for (int i : v) cout << i << " ";  
    cout << endl;  
  
    v.pop_back(); // delete  
    cout << "Vector after pop_back: ";  
    for (int i : v) cout << i << " ";  
    cout << endl;  
  
    // List operations  
    list<int> l = {1, 2, 3};  
    l.push_back(4); // insert  
    l.push_front(0); // insert at front  
    cout << "List elements: ";  
    for (int i : l) cout << i << " ";  
    cout << endl;  
  
    l.pop_back(); // delete  
    l.pop_front(); // delete front  
    cout << "List after deletion: ";  
    for (int i : l) cout << i << " ";  
    cout << endl;
```



A D I T Y A
U N I V E R S I T Y

Date:

Roll No.:

2	4	B	1	1	A	1	2	3	3
---	---	---	---	---	---	---	---	---	---

```
cout << "M.Dileep" << endl;  
return 0;  
}
```

Output:

```
Vector elements: 10 20 30 40  
Vector after pop_back: 10 20 30  
List elements: 0 1 2 3 4  
List after deletion: 1 2 3  
M.Dileep  
  
=== Code Execution Successful ===
```

CO Mapped: CO4

POs Mapped: PO1 , PO2 , PO3 , PO4 , PO5 , PO9 , PO11

PSOs Mapped: PSO1

Date:

Roll No.:

2	4	B	1	1	A	I	2	3	3
---	---	---	---	---	---	---	---	---	---

Program No: 12(b)

Implement Deque in C++ and demonstrate basic operations.

Aim: Implement Deque in C++ and demonstrate basic operations.

Description:

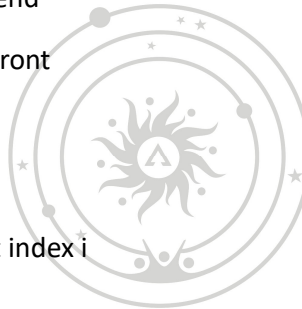
This program demonstrates deque (double-ended queue) operations in C++ such as insertion at both ends, deletion, and traversal.

Syntax:

```
#include <deque>

deque<type> dq;    // declare deque

dq.push_back(value); // insert at end
dq.push_front(value); // insert at front
dq.pop_back();      // delete last
dq.pop_front();     // delete first
dq[i];              // access element at index i
```



Program:

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

int main() {
    deque<int> dq;

    dq.push_back(10);
    dq.push_front(5);
    dq.push_back(20);
```

A D I T Y A
U N I V E R S I T Y

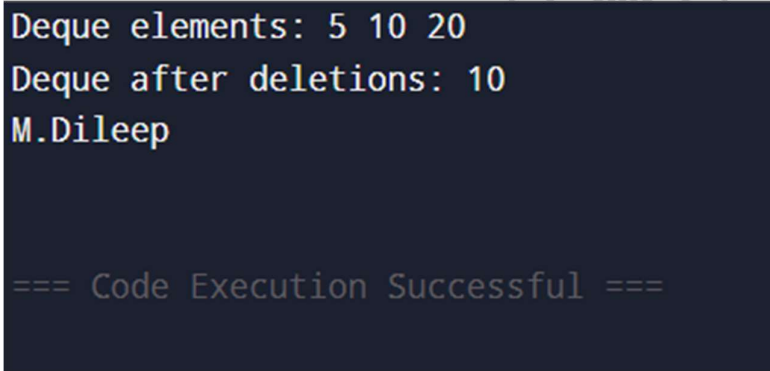
Date:

Roll No.:

2	4	B	1	1	A	1	2	3	3
---	---	---	---	---	---	---	---	---	---

```
cout << "Deque elements: ";  
for (int i : dq) cout << i << " ";  
cout << endl;  
  
dq.pop_front(); // delete from front  
dq.pop_back(); // delete from back  
cout << "Deque after deletions: ";  
for (int i : dq) cout << i << " ";  
cout << endl;  
cout << "M.Dileep" << endl;  
return 0;  
}
```

Output:



```
Deque elements: 5 10 20  
Deque after deletions: 10  
M.Dileep  
  
=== Code Execution Successful ===
```

CO Mapped: CO4

POs Mapped: PO1 , PO2 , PO3 , PO4 , PO5 , PO9 , PO11

PSOs Mapped: PSO1

Date:

Roll No.:

2	4	B	1	1	A	I	2	3	3
---	---	---	---	---	---	---	---	---	---

Program No: 12(c)

Implement Map and demonstrate operations such as insertion, deletion, access, and searching.

Aim: Implement Map and demonstrate operations such as insertion, deletion, access, and searching.

Description:

This program demonstrates the use of map in C++ STL. It covers inserting key-value pairs, deleting elements, accessing values, and searching for keys.

Syntax:

```
#include <map>

map<key_type, value_type> m; // declare map

m[key] = value;           // insert/update
m.insert({key, value});   // insert using pair
m.erase(key);            // delete key-value pair
m.find(key);              // search for key
m[key];                   // access value
```

Program:

```
#include <iostream>
#include <map>

using namespace std;

int main() {
    map<int, string> m;

    // Insertion
    m[1] = "Nithin";
    m[2] = "Sarvan";
    m.insert({3, "Akhil"});
```

Date:

Roll No.:

2	4	B	1	1	A	I	2	3	3
---	---	---	---	---	---	---	---	---	---

```
cout << "Map elements:" << endl;
for (auto &p : m) {
    cout << p.first << " -> " << p.second << endl;
}
// Access
cout << "Access key 2: " << m[2] << endl;
// Deletion
m.erase(1);
cout << "Map after deletion:" << endl;
for (auto &p : m) {
    cout << p.first << " -> " << p.second << endl;
}
// Searching
int key = 3;
if (m.find(key) != m.end()) cout << "Key " << key << " found with value: " << m[key] << endl;
cout << "M.Dileep" << endl;
return 0;
}
```



A D I T Y A
U N I V E R S I T Y

Output:

Date:

Roll No.:

2	4	B	1	1	A	1	2	3	3
---	---	---	---	---	---	---	---	---	---

```
Map elements:
1 -> Nithin
2 -> Sarvan
3 -> Akhil
Access key 2: Sarvan
Map after deletion:
2 -> Sarvan
3 -> Akhil
Key 3 found with value: Akhil
M.Dileep
```

```
=== Code Execution Successful ===
```

CO Mapped: CO4

POs Mapped: PO1 , PO2 , PO3 , PO4 , PO5 , PO9 , PO11

PSOs Mapped: PSO1


A D I T Y A
U N I V E R S I T Y