```
//-----------------------VECTORS-------------------------------------------------------

/*
VECTOR -> Vector is a dynamic sized array which can grow & shrink it's size which makes
          it versatile and efficient data structure for storing & manipulating sequence
          of elements.

          Vector also stores its elements in continous memory blocks just like array so
          that we cam access each element in the vector by using iterator or loop.



VECTOR STL -> STL provides offer a collection of functions that aftercommon data structures
              and algorithms to make programming more efficient and convenient.

              It means it provides an implementation for data structures like array. Using this
              implementation, we need not to create an array by our own so that the user or developer
              can efficiently write code by focusing more on the logics rather than creating array and
              defining its size.

              This implementation will work same arrays

              Talking about arrays, STL provides a way that how to create arrays how to use arrays etc and by using
              this way or implementation or the functions provided by the STL arrays, we will get the same behaviour
              as array.

             So, we can say that the implementation to create dynamic arrays is written in STL
             (Standard Template Library).

    FEATURES OF VECTOR -->
    1[ Continous memory allocation
    2] Dynamic sizing
    3] Automatic reallocation -> accomodate new elements
    4] Size and capacity
    5] Array like access -> using square brackets[] and can also access by at() member function.
*/

// VECTOR PRINTING

#include<iostream>
#include<vector>
using namespace std;
void print(vector<int>v)
{
     for(int i = 0; i < v.size(); i++)
```

```
     {
         cout << v[i] << " ";
     }
     cout << endl;
 }
int main()
{
     vector<int>v;

    // To push or insert the elements in vector
    v.push_back(1);
    v.push_back(2);
    v.push_back(3);
    v.push_back(4);
    v.push_back(5);

    print(v);  // function call

    cout << "Capacity of vector is: "<< v.capacity() << endl;
    cout << "Size of Vector is: "<< v.size() << endl;

    cout << v.front() << endl;  // used to find the front element
    cout << v.back() << endl;  // used to find the back element
    return 0;
}

// TO FIND CAPACITY AND SIZE OF VECTOR -->

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

int main()
{
       vector<int>v;

       while(1)
       {
              int data;
              cout << "Enter your data in the vector" << endl;
              cin >> data;

              v.push_back(data);
              cout << "Capacity: " << v.capacity() << " " << "Size: " << v.size() << endl;
       }
```

```
}

// NOTE: Here when the capacity get full the size is getting doubled.

// TO PUSH BACK AND POP UP THE VECTOR  -->

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

void printvec(vector<int>v)
{
	cout << "Print the vector" << endl;
	int size = v.size();
	for(int i = 0;i<size;i++)
	{
		cout << v[i] << " ";
	}
	cout << endl;
}
int main()
{
	vector<int>v;

	v.push_back(1);
	v.push_back(2);
	v.push_back(3);
	v.push_back(4);
	v.push_back(5);
	printvec(v);

	v.pop_back();
	printvec(v);

	v.pop_back();
	printvec(v);

	return 0;

}

// TO TAKE INPUT FROM USER FOR VECTOR SIZE AND INSERTINF MORE ELEMENTS WITHOUT ASKING USER

#include<iostream>
#include<vector>
```

```
using namespace std;

void print(vector<int>v)
{
	cout << "Print the vector" << endl;
	int size = v.size();
	for(int i = 0;i<size;i++)
	{
		cout << v[i] << " ";
	}
	cout << endl;
}
int main()
{
	vector<int>v;
	int n;
	cin >> n;
	for(int i=0;i<n;i++)
	{
		int d;
		cin >> d;
		v.push_back(d);
	}
	print(v);

	cout <<"Capacity before pushing elements: "<< v.capacity() << endl;
	cout << "Size before pushing elements: " << v.size() << endl;

	// for push back more elements without asking user

	for(int i=0;i<5;i++)
	{
		v.push_back(6);
	}
	print(v);
	cout <<"Capacity after pushing elements: "<< v.capacity() << endl;
	cout << "Size after pushing elements: " << v.size() << endl;
}

//-------------------------------------------------------------------------------

// 2D Vector

/* DECLARATION -> vector<vector<int>>v;
```

INTIALIZATION ->

1] vector<vector<int>>v { {1,2,3},{4,5,6}};

2] vector<vector<int>>v = {{1,2,3},{4,5,6}};

3] vector<vector<int>>v(5,vector<int>(5,2));

```
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
```

ACCESS OF ELEMENTS -> arr[i][j]

BEHIND THE SCENE IN 2D ARRAY ->

There is no allocation of 2D space in memory for 2D arrays.
They are just like linear arrayvin continous memory blocks.

MAPPING OF 1D ARRAY IN MEMORY IS DONE BY USING FORMULA ->

c × i + j

c is columns total
i is row index
j is column index

NOTE: when an array i.e 2D is initialized , it is mandatory
to give the size of columns bcz mapping of 2D array in
linear arrays requires size of columns.

Also when 2D array is passed , the size of columns
must be told.

Its optional to give the size of rows in 2D array.
*/

```cpp
#include<iostream>
#include<vector>
using namespace std;
 int main()
 {
     vector<vector<int>>v = {{1,2,3},{4,5,6},{7,8,9}};

     for(int i = 0; i < v.size(); i++)
     {
         for(int j = 0; j < v[i].size(); j++)
         {
             cout << v[i][j] << " ";
         }
         cout << endl;
     }
 }

 // v.size()  --> size of rows
 // v[i].size()  --> size of compl

 //-------------------------------------------------------------------

 // JOGGED ARRAY  -> In which no of col may vary.It can be done by making
 // 2D vector and push back 1D vector inside that 2D vector.



#include<iostream>
#include<vector>
using namespace std;
 int main()
 {
     vector<vector<int>>v;  // 2D vector

     vector<int>v1(1,1); // 1D vector
     vector<int>v2(2,2); //1D vector
     vector<int>v3(3,3);  //1D vector

     // push_back 1d vec to 2d vec
     v.push_back(v1);
     v.push_back(v2);
     v.push_back(v3);
```

```
    for(int i = 0; i < v.size(); i++)
    {
        for(int j = 0; j < v[i].size(); j++)
        {
            cout << v[i][j] << " ";
        }
        cout << endl;
    }
 }

//---------------------------------------------------------------------


// STATIC MEMORY ALLOCATION IN ARRAY ---->

#include<iostream>
using namespace std;
void func(int arr[] , int n)
{
       for(int i = 0; i < n; i++)
       {
              cout << arr[i] << " ";
       }
}
int main()
{
       int arr[5] = {1,2,3,4,5};
       int n = 5;
       func(arr,n);
       return 0;
}

// ---------------------------------------------

// DYNAMIC MEMORY ALLOCATION IN ARRAY -->

#include<iostream>
using namespace std;
void func(int arr[] , int n)
{
       for(int i = 0; i < n; i++)
       {
              cout << arr[i] << " ";
       }
}
```

```
int main()
{
	int n;
	cout << "Enter the size of array: ";
	cin >> n;
	int *arr = new int[n]; // dynamic

	for(int i = 0;i < n; i++)
	{
		int data;
		cin >> data;
		arr[i] = data;
	}

	func(arr,n);
	return 0;
}

/*
suppose we have taken input value 5 to add 5 elements in the array. But now
requirement increaes and we want to add some more elements in the array. we cant
simply do that

The solution for this is STL which will define the implementation of vector
Data Structure.

In vector Data structure we need not to tell the size of vector just keep inserting
the required no of elements in vector.
*/
```