C++ STL

- using namespace std
  OR
  std :: cin >> a;

#include <utility>

Pairs → Store 2 Value at a time

$$\underbrace{\text{pair <int, int>}}_{\text{Datatype}}\ p = \{\underbrace{1,\ 2}_{\text{Value}}\}$$

- To access it
  p.first → 1
  p.second → 2

Store 3 Values.

pair <int, pair <int, int>> p = {1, {3, 4}}

- To access it
  p.first → 1
  p.second.first → 3
  p.second.second → 4

Date ________

Pair Array

pair <int, int> arr[] = $\{ \underbrace{\{1,2\}}_{0}, \underbrace{\{3,4\}}_{1}, \underbrace{\{5,6\}}_{2} \}$.

arr[1].second = 4

Vectors → is a dynamic Array like Data structure that is part of STL

```
vector <int> v;

    v.push_back(1);
    v.emplace_back(2);
```

List → is another type of Sequential container provided by the STL.

• While a 'vector' is typically implemented as a dynamic array,
↳ A list is usually implemented as a doubly-linked list. Each element of a 'list' contains a pointer to the next and previous elements, allowing for efficient insertion and deletion at any position.

↳ Not support accessing elements by index.

Date ____________

Deque

↳ (double-ended Queue) is another container provided by the C++ Standard Library.

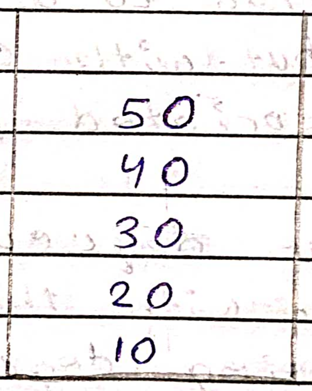
↳ It's a sequence container that allows insertion and deletion of elements at both ends efficiently.

Stack

↳ is data structure that follow the Last in, First Out (LIFO) principle.

- ↳ Push
- ↳ Pop
- ↳ Top
- ↳ Empty
- ↳ Size

→ All are O(1) Time complexity



```
stack <int> st1 ;
stack <int> st2 ;

st1.swap(st2) ;
```

Queue

↳ is a linear data structure that follows the first in, first out (FIFO) principle.

↳ Elements are Added at the rear
↳ enqueue

↳ Removed from the front.
↳ dequeue

10 20 30 40
↳ front ↳ Rear

Priority Queue

↳ is an abstract data type similar to a regular queue or stack, but with each element having an associated priority.

↳ Elements are dequeued in order of priority, with the highest priority elements being dequeued first.

• Minimum Priority Queue [Min Heap]

push
pop] → log n

Top → O(1)

Date ________

Sets

↳ are associative containers that store unique elements following a specific order

Everything happen in → log N

Multi-Set

↳ is an associative container that allows multiple elements with the same value to be stored.
↳ It is similar to a "set", but unlike a set, it allows duplicate elements.

Unordered Set

↳ is an associative container that contains a set of unique objects. It is similar to "set", but the elements are not ordered. Instead elements are organized into buckets based on their hash values for fast retrieval.

- lower bound / upper bound Not Works

Mostly → O(1)

Onces in a Blue Moon → O(N)

Date ______

Map

containes that stores elements formed by a combination of a key and a Value, where each key is unique.

↳ referred to as a dictionary or an associative array.

↳ sorted by their keys.

Multi-map :

↳ Same as Map
↳ it can store Duplicate key
↳ map[key] cannot used.
↳ sorted

Unordered-map

↳ Same as above
↳ But Not Sorted

Algorithms

↳ sort
↳ comp (My way)

• __builtin_popcount
↳ return no. of Set bits (1)
↳ 6 → 110 → (2)
↳ 7 → (3)

Date

- permutation → 123
  132
  213
  231
  312
  321

```
string s = "123"

next_permutation (s.begin(), s.end())

    # Start from the sorted
```

- max_element (a, a+n);
- min_element (a, a+n);