Standard Template Library

Topics to cover in chronological order:

1. Introduction to Standard Template Library:

- C++/Python Joke
- What is our role as a programmer?
- How STL helps us solve this problem:
  The C++ STL (Standard Template Library) is a powerful set of C++ template classes to provide general-purpose classes and functions with templates that implement many popular and commonly used algorithms and data structures like vectors, lists, queues, and stacks.

2. Introduction to C++:

- Start by asking how many people know C and C++
- All syntax of C works in C++
- Explain alternative of C syntax in C++

  a. The `#include<bits/stdc++.h>`

It is basically a header file that includes every standard library. In programming contests, using this file is a good idea, when you want to reduce the time wasted in doing chores; especially when your rank is time sensitive.

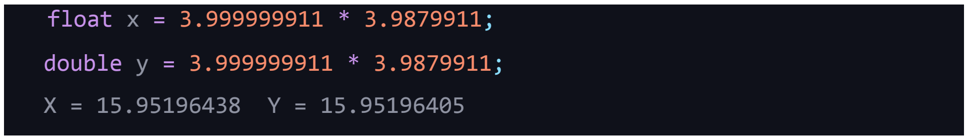
b. The `using namespace std;`

Namespaces are commonly structured as hierarchies to allow reuse of names in different contexts. As an analogy, consider a system of naming of people where each person has a given name, as well as a family name shared with their relatives. If the first names of family members are unique only within each family, then each person can be uniquely identified by the combination of first name and family name; there is only one Jane Doe, though there may be many Janes. Within the namespace of the Doe family, just "Jane" suffices to unambiguously designate this person, while within the "global" namespace of all people, the full name must be used.

c. Data Types in C++

- Integer
- Float
- Double

  Double vs Float:



```
 float x = 3.999999911 * 3.9879911;
double y = 3.999999911 * 3.9879911;
X = 15.95196438  Y = 15.95196405
```

- Character
- String
- Boolean

d. Input/Output in C++

- cout
- << (Insertion operator)
- cin
- >> (Extraction operator)

e. If/Else statements

f. Loops

- `for(int i=0;i<n;i++)`
- `while(t>0)`

PRACTICE PROBLEM:
Given an integer ‘n’, print all squares for integers from 1 to n

INput:2
OUput:1 4

INput:3
OUput:1 4 9

INput:7
OUput:1 4 9 16 25 36 49

3. Arrays in C++:

- Primitive way of declaring arrays

```
int arr[] = {1,2,3,4,5};


    int arr[5];
    arr[0] = 1;
    arr[1] = 2;
    arr[2] = 3;
```

- Array Container in STL

```
array<int,5>arr = {1,2,3,4,5};
```

- Inbuilt Functions of arrays
    a. arr.size()
    b. arr.front()
    c. arr.back()
    d. arr.fill(10)

- Iterators
    a. .begin()
       begin() function is used to return an iterator pointing to the first element of the container
    b. .end()
       end() function is used to return an iterator pointing to next to last element of

the container

c. `.rbegin()`

rbegin() function is used to return an iterator pointing to the last element of the Container

d. `.rend()`

rend() function is used to return an iterator pointing next to the first element of the Container

- Functions

  a. `sort(arr.begin(),arr.end())`

  sorts the elements in the array

  b. `reverse(arr.begin(),arr.end())`

  Reverses the order of elements in the array

  c. `is_sorted(arr)`

  Checks if the elements in the array are sorted in ascending order

  d. `max_element(arr.begin(),arr.end())`

  Returns a pointer to the largest element in the array

  e. `min_element(arr.begin(),arr.end())`

  Returns a pointer to the smallest element in the array

  f. `binary_search(arr.begin(),arr.end(),x)`

  Returns if an element is present in the array

PRACTICE PROBLEM:

a. ***Ram was going home when he walked over a magical box and an envelope. He opened the envelope and it read the following :***

   “ This box contains the rarest diamonds in the world, but there is a catch. To open this box, you need to find it’s password. The password happens to be the largest element in the array given below. Good luck “

   Help Ram find the password and open the magical box.

   Input:
   The first line will contain an integer ‘n’ , the size of the array.
   The next line will contain n space separated integers a1, a2 ,.... an
   1 <= n <= 1e5
   1<= ai <= 1e5

   Output:
   Output the largest integer in the array

Example case:

Input:
5
1 2 3 4 5

Output:
5

INput:
5
44 66 55 11 36

OUput:
66

INput:2
1
999
OUput:
999

INput:
10
123 432 993 8866 1 12345 764 432 10 8
OUput:
12345

4. Vectors:

- Point of difficulty with working with arrays
- Show how vectors work efficiently with dynamic memory allocation
- Declaring vectors in C++

```
vector<int>vec1 = {1,2,3,4,5};
vector<int>vec2(5);
vector<int>vec3(5,10);
```

- `Methods on vectors`
    a. `.push_back(x)`
       Adds an element x to the end of vector and increases the size of the vector by 1
    b. `.pop_back()`
       Removes the last element from the vector and reduces the size of the vector by 1
    c. `.insert(vec.begin(),{1,2,3,4,5})`
       Adds elements to a specific position in the vector
    d. `.assign(n,x)`
    e. `.resize(n,x)`

- `Useful Functions`
    a. `accumulate(vec.begin(),vec.end(),x)`
       Returns the sum of the elements in the given range starting from x
    b. `partial_sum(vec.begin(),vec.end(),prefix.begin())`
       Assigns values to elements in the vector prefix that are sums of partial ranges of the vector

- `Show how to iterate through a vector without using a for each loop`

PRACTICE PROBLEM:

a. ***Dimitry was walking past a garden when he saw 'n' strings standing one behind another in a queue in front of an ice cream stall. At that instant, it is announced that the 'k'th string from the back of the queue has won a jackpot of 1000 ice-creams.***
Help Dimitry find who won that jackpot

Input:
The first line will contain 2 integers 'n' and 'k' , the size of the array.
The next line will contain n space separated strings a1, a2 ,.... an
1 <= k <= n <= 1e5
1<= |ai | <= 1e5

Output:
Output the string who has one the jackpot

Example case:

Input:
5 2
Bob jack jill Gill moss

Output:
Gill

```
INput:
10 1
Gaurish Jay Hussain Arya Pranay Pargat Devansh Nishit Neelansh Tanvi
OUput:
Tanvi

INput:
5 3
Jack Jill Bill Kilt Molova
OUput:
Bill
```

5. Pairs:

- Explain the importance of having pairs. (let's say to store and manipulate coordinates)

- Declaring pairs

```
pair<int,int>coordinates = {1,2};
pair<string,int>StudentMarks ={"Raj",55};
```

- Accessing elements from a pair
  a. .first
     The first element in the pair
  b. .second
     The second element in the pair

- Accessing Elements in a vector/ array of pairs

```
   pair<int,int> coord[5];
   coord[0] = {1,1};

   coord[1].first = 1;
   coord[1].second = 2;

   for(int i=0;i<5;i++){
     cout<<coord[i].first<<" "<<coord[i].second<<"\n";
```

```
    }
```

PRACTICE PROBLEMS:

a. ***Gordon the dragon has suddenly spawned. He is threatening to kill you but you are able to bargain him to a pact. According to the pact, Gordon will give you a challenge, if you're able to solve it you win and Gordon will disappear , otherwise he will kill you.***
The problem is as follows:
Gordon gives you 'n' pairs of integers (a1,b1), (a2,b2)....(an,bn). You need to find the number of pairs such that parity of ai == parity of bi.

Parity of an integer is its attribute of being odd or even.

Input:
The first line will contain an integer 'n' , the size of the array.
The next n lines contain 2 integers each, ai and bi

1 <= n <= 1e5
1<= ai , bi <= 1e5

Output:
Output the number of pairs with the same parity

Example case 1:
Input:
5
1 121
20 33
41 76
90 68
3 39

Output:
3

b. ***Krokamonstor has ‘n’ dolls standing in a line such that the ai th doll is in the ith position in the line. The aith doll has a strength ‘ai’. Krokamonstor wants to rearrange these dolls in such a way that if the position of a doll is p, then no doll in between 1<= r < p should have a strength greater than doll p. Print the original index of the dolls in which they were standing in order of the new line.***

Input:
The first line will contain an integer ‘n’ , the size of the array.
The next line will contain n space separated integers s1, s2 ,.... Sn, i.e. the strength of the ith doll

1 <= n <= 1e5
1<= si <= 1e5

Output:
Output the indexes of the dolls when placed in optimal order

Example case 1:
Input:
5
108 46 72 93 22

Output:
5 2 3 4 1

6. Lists:

- Explain problem with vectors/arrays (what if continuous memory location is not present)

- How does a linked list solve this problem?
  https://medium.com/basecs/whats-a-linked-list-anyway-part-1-d8b7e6508b9d

- Declaring a linked list

```
    list<int>list0;
    list<int>list1(6,0);
    list<int>list2 = {1,2,3,4,5};
```

- Methods on lists
    a. .push_back(x)
       Adds an element x to the end of list
    b. .pop_back()
       Removes the last element from the list
    f. .push_front(x)
       Adds an element x to the front of list
    g. .pop_front()
       Removes the first element from the list

- Iterating through a list/disadvantages of lists

```
    list<int>list2 = {1,2,3,4,5};

    for(auto i : list2){
      cout<<i<<" ";
    }
```

PRACTICE PROBLEMS:

Mr. Vibhuti Bonbon is a weird man.He wants to keep adding elements to an empty list either at the end according to his mood. Given n integers and his mood (0==insert at front, 1==insert at back) print the correct order of the list

Input:
The first line will contain an integer 'n' , the size of the array.
The next n line will contain 2 space separated integers ai and ki, where ai is the element and ki is his mood

1 <= n <= 1e5
1<= ai <= 1e5
0<= ki <= 1

Output:
Output the indexes of the dolls when placed in optimal order

Example case 1:
Input:

5
5 0
3 0
2 1
4 1
6 0

Output:
6 3 5 2 4

7. Sets:

- Explain the mathematical meaning of sets

- Declaring sets

```
set<int>set1 = {1,2,3,4,5};


set<int>set2;
set2.insert(1);
set2.insert(1);
set2.insert(2);
```

- Set methods
  a. .insert(x)
     This inserts an element x in the set
  b. .size()
     This returns the size of the set
  c. .clear()
     This clears the entire set

- Accessing elements in a set

```
set<int>set1 = {1,2,3,4,5};



for(auto i : set1){
  cout<<i << " ";
}
```

- Remove elements from a set

```
set<int>set1 = {1,2,3,4,5};
set1.erase(5);
```

PRACTICE PROBLEMS:

a. ***Mathematician Ramanujan wants you to solve a problem. Given an array of n integers, print all the unique numbers present in the array.***

   Input:
   The first line will contain an integer 'n' , the size of the array.
   The next line will contain n space separated integers a1, a2 ,.... an
   1 <= n <= 1e5
   1<= ai <= 1e5

   Output:
   Print all unique numbers from the array

   Example case 1:
   Input:
   10
   1 3 5 6 8 8 1 2 1 1

   Output:
   1 2 3 5 6 8

8. Maps:

- Explain necessity of maps(if you want to have data corresponding to some values, let's say frequency of an element in an array)

- Explain concept of key value pairs

- Declaring Maps

```
map<int,int>map1;
unordered_map<int,int>map2;
```

- Adding values to a map. (use it just like array :))

```
map1[3]++;
map1[4]++;
```

```
map<string,int>map3;
map3["Bob"]++;
```

- Iterating through a map

```
    for(auto i : map3){
      cout<<i.first<<" "<<i.second<<"\n";
    }
```

PRACTICE PROBLEMS:

a. ***Grandmaster Nenu has a challenge for you. Nenu gives you an array of 'n' integers and asks you to find the integer which has the highest frequency(occurrences) in the array. Complete Grandmaster Nenu's challenge. If there are multiple such integers, print any of the integers that satisfies the given condition.***

   Input:
   The first line will contain an integer 'n' , the size of the array.
   The next line will contain n space separated integers a1, a2 ,.... an
   1 <= n <= 1e5
   1<= ai <= 1e5

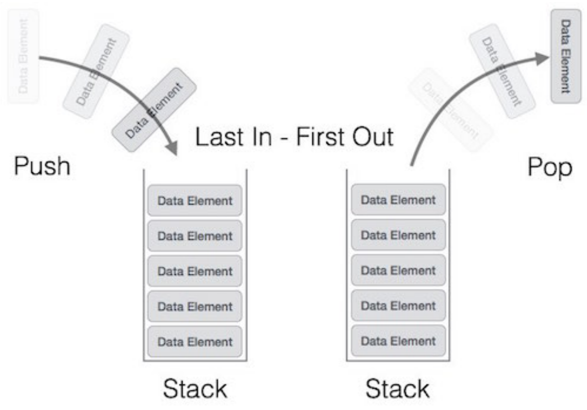
   Output:
   Print the integer from the array which has with the highest frequency

   Example case 1:
   Input:
   10
   1 3 5 6 8 8 1 2 1 1

   Output:
   1

9. Stacks:

- Explain stacks with diagram



- Declaring a stack

```
    stack<int>stack1;
```

- .push()

  This adds an element from the back of the stack

```
    stack<int>stack1;
    stack1.push(2);
    stack1.push(3);
    stack1.push(5);
```

- .pop()

  This removes an element from the back of the stack

```
    stack1.pop();
```

- .top()

  This returns the element at the back of the stack

```
    cout << stack1.top();
```

- Iterating through a stack

```
    stack<int>stack1;


    stack1.push(3);
    stack1.push(2);
    stack1.push(1);
    stack1.push(5);



    while (stack1.empty()!=true){
      cout<< stack1.top()<<" ";
      stack1.pop();
    }
```

PRACTICE PROBLEMS:

a. ***You have been lying on your bed for too long and your mom is not happy about it. So she decided to give you a task. She gives you an array of n integers and she wants you to reverse the array. But there is a catch, you can only use a STACK to reverse the array.***

   Input:
   The first line will contain an integer 'n' , the size of the array.
   The next line will contain n space separated integers a1, a2 ,.... an
   1 <= n <= 1e5
   1<= ai <= 1e5

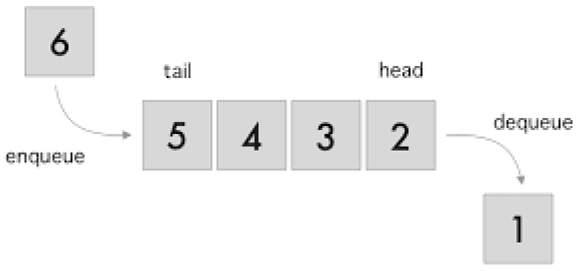
   Output:
   n space separated integers

   Example case 1:
   Input:
   5
   1 3 5 6 8

   Output:
   8 6 5 3 1

10. Queues :

- Explain queues with real life examples. Call people for demonstration if there is space in lab



- Declaring a queue

```
    queue<int>q;
```

- .push()

This adds an element from the back of the queue

```
    queue<int>q;


    q.push(2);
    q.push(3);
    q.push(5);
    q.push(1);
```

- .pop()

This removes an element from the front of the queue

```
    q.pop();
```

- .front() and .back()

This returns the first and the last element from the queue respectively

```
    cout << q.front() << " " << q.back() << "\n";
```

- Iterating through a queue

```
    queue<int>q;


    q.push(2);
    q.push(3);
    q.push(5);
    q.push(1);



    while(q.empty()==false){
      cout<<q.front()<<" ";
      q.pop();
    }
```

PRACTICE PROBLEMS:

TODO: frame question. Increase difficulty.

Add 100 on 'yes' remove 50 on 'no'. keep track of all operations. Using queue.

Input:

y n y n n y

Output:

100 50 150 100 50 150

11.Priority Queues:

- Priority queues are a type of container adapters, specifically designed such that the first element of the queue is the greatest of all elements in the queue and elements are in nonincreasing order

- Declaring a max priority queue

```
priority_queue<int>pq;
```

- .push(x)

  This pushes an element x to the priority queue. Then internally the priority queue reorders the elements in such a way such that the largest element is at the front.

- .pop()

  This removes the last/smallest element from the priority queue

- .top()

  This returns the largest/topmost element of the priority queue

- .empty()

  Returns true if the priority queue is empty

- Implementation of priority queue

```
    priority_queue<int>pq;


    pq.push(1);
    pq.push(2);
    pq.push(3);
    pq.push(4);
```

```
while(pq.empty()==false){
  cout<<pq.top()<<" ";
  pq.pop();
}
```

PRACTICE PROBLEMS:

b. ***Your mother is still not happy with you. So she decided to give you another task. She gives you an array of n integers and she wants you to sort the array in descending order. But there is a catch, you can only use a QUEUE for this task.***

Input:
The first line will contain an integer 'n' , the size of the array.
The next line will contain n space separated integers a1, a2 ,.... an
1 <= n <= 1e5
1<= ai <= 1e5

Output:
Sort the array in reverse order.

Example case 1:
Input:
5
100 30 -5 16 0

Output:
100 30 16 0 -5

12. Nested Containers

- Make them understand how nested containers work using PPAP reference

- While using container methods, first preference is given to the parent container

- Implementation

```
    map<int,vector<int>>nest;



   for(int i=0;i<10;i++){
      for(int j=0;j<5;j++){
        nest[i].push_back(j+1);
      }
   }

   for(int i=0;i<10;i++){
     cout<<"key "<<i<<": ";

     for(auto k : nest[i]){

       cout<<k<<" ";

     }
     cout<<"\n";
   }
```

The End...