IEEE CP SMP 2018
Assignment 1
Topic: STL (Standard Template Library)

Write programs in C++ to get familiarised with the following containers in C++ STL. Get familiarised with how the data structures work, how they are used , and what they do. For each question start with an empty container at the beginning of the queries, and assume all inputs are valid. Do not print unnecessary statements (like “Enter a number”) to the output stream, as this would lead to a WA (Wrong answer) on any online judge. Write neat code, with proper indentation. **Use the inbuilt functions** for all the operations, do not implement these functionalities on your own. **Send a pull request** to the GitHub repository https://github.com/IEEE-NITK/SummerProjects18 by maintaining your fork. **Name the folder with your name**, and check out the “Sample” folder to see how your files must be structured and named.

DEADLINE : 11:59pm, May 9th, Wednesday

Input Format
First line contains Q - number of queries, which is followed by Q lines 1 for each query. The inputs for the query are described in each question.

Output Format
Print the answer to each query on a new line.

1. Stacks

Elements of vector are of type int. Queries are of type:
1 : Print the top most element
2 x : Push x into the stack
3 : Pop the top most element
4 : Print size of stack

Example:
Input:

```
7          -> 7 queries
2 5        -> Query type 2, x = 5
2 7        -> Query type 2, x = 7
1          -> Query type 1
4          -> Query type 4
3          -> Query type 3
4          -> Query type 4
1          -> Query type 1
```

Output:

```
7
2
1
5
```

2. Queues

Elements of vector are of type int. Queries are of type:
1 : Print the front most element
2 : Print the rear most element
3 x : Push x into the queue
4 : Pop the front most element
5 : Print size of queue

3. Vectors

Elements of vector are of type int. Queries are of type:
1 x : Push x at the end of the list
2 x y : Update the element at position x (starting from 0) to y
3 x : “Yes” if x is present in the list, “No” otherwise
4 x : Remove x from the list (first occurrence)
5 : Print size of list
6 : Sort the entire list
7 : Print the entire vector in the format “a1 a2 a3 … an” (space seperated).

4. Arrays

Elements of array are of type int. Keep the size of the array as 1000, with all values initialized to 0 in the beginning (use “memset()”). Queries are of type:
1 x y : Set element at position x to y (starting from index 0)
2 : Sort the elements (ascending order)
3 x : Print position of lowerbound of x in the array (assume array is sorted)
4 x : Print position of upperbound of x in the array (assume array is sorted)
5 : Update the array to its next permutation and print the array
6 : Update the array to its previous permutation and print the array

5. Pair

Elements of vector are of type <int, int>. Keep an array of them of size 1000, with all elements initialized to (0, 0). Queries are of type:
1 x y z : Set element at position to (y, z)
2 1 : Sort by giving preference to first element over second element
2 2 : Sort by giving preference to second element over second element
3 : Print the array in the format "(x1, y1) (x2, y2) (x3, y3) ... (xn, yn) "

6. Priority Queue

Elements of are of type pair< int, pair<int, int> >. Try to see how the priority is getting assigned in this case. Queries are of type:
1 x y z : Insert (x, (y, z))
2 : Print the top most element in the format “(x, (y, z))”
3 : Pop the top most element
4 : Print the size

7. Map

String maps to int. Queries are of type:
1 x y : Map x to y (x is string, y is int)
2 x : Print value mapping to string x
3 x : Check if x is present as a key in the map
4 : Print the entire map using iterators in the format "(str, val) (str2, val2) ..."

8. Set

Elements of vector are of type int. Queries are of type:
1 x : Insert x into the set
2 : Print size
3 : Remove first element
4 : Print first element
5 : Print entire list using iterators
6 : Print last element
7 x : ”Yes” if x is in the set, else “No”
8 x : Remove x from the set

9. MultiSet

Elements of are of type pair< int, pair<int, int> >. Try to differentiate between this and the priority queue. Queries are of type:
1 x y z : Insert (x, (y, z))
2 : Print the top most element in the format “(x, (y, z))”
3 : Pop the top most element
4 : Print the entire priority queue using an iterator
5 : Print the size

10. Double Ended Queue

Elements of vector are of type int. Queries are of type:
1 : Print the front most element
2 : Print the rear most element
3 x : Push x into front end of the queue
4 x : Push x into rear end of the queue
5 : Pop the front most element
6 : Pop the rear most element
7 : Print size of queue