# SORT A STACK

There can be many solutions to this question. But let us focus on two methods for now , one by just traversing the stack and comparing each element with all the elements , and the other through recursion .Lets discuss both of them one by one.

## SOLUTION 1

CODE :

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

using namespace std;

class SortedStack {

 public:

  stack<int> s;

  void sort();

};



void printStack(stack<int> s) {

  while (!s.empty()) {

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

```
    s.pop();
  }
  cout << endl;
}
int main() {
  int t;
  cin >> t;
  while (t--) {
    SortedStack* ss = new SortedStack();
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
      int k;
      cin >> k;
      ss->sort();
      printStack(ss->s);
    }
  }
  return 0;
}


void SortedStack ::sort() {
  stack<int> sorted;
```

```
  int t;
  while (!s.empty()) {
    t = s.top();
    s.pop();
    if (sorted.empty()) {
      sorted.push(t);

    } else if (sorted.top() < t) {
      while (!sorted.empty() && sorted.top() < t) {
        s.push(sorted.top());
        sorted.pop();
      }
      sorted.push(t);
    } else
      sorted.push(t);
  }
  while (!sorted.empty()) {
    s.push(sorted.top());
    sorted.pop();
  }
}
```

}

# SOLUTION 2

## CODE:

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

using namespace std;

class SortedStack {
  public: stack < int > s;void sort();
};

void printStack(stack < int > s) {
  while (!s.empty()) {
    cout << s.top();
    s.pop();
  }
  cout << endl;
}
int main() {
  int t;
  cin >> t;
  while (t--) {
    SortedStack * ss = new SortedStack();
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
```

```
      int k;
      cin >> k;
      ss -> sort();
      printStack(ss -> s);
    }
  }
  return 0;
}

void insert(stack < int > & s, int temp) {
  if (s.size() == 0 || s.top() <= temp) {
    s.push(temp);
    return;

  } else {
    int x = s.top();
    s.pop();
    insert(s, temp);
    s.push(x);
  }
}
void SortedStack::sort() {
  if (s.size() == 1)
    return;
  int temp = s.top();
  s.pop();
  sort(s);
  insert(s, temp);
}
```