1. **Function to sort an array using insertion sort**

void insertionSort(int arr[], int n)

{

    int i, key, j;

    for (i = 1; i < n; i++)

    {

        key = arr[i];

        j = i - 1;

**Move elements of arr[0..i-1], that are greater than key, to one position ahead**

        of their current position \*/

        while (j >= 0 && arr[j] > key)

        {

            arr[j + 1] = arr[j];

            j = j - 1;

        }

        arr[j + 1] = key;

    }

}

**2. C++ program to keep track of maximum**

#include <bits/stdc++.h>

using namespace std;

class StackWithMax

{

stack<int> mainStack;

stack<int> trackStack;

public:

void push(int x)

{

mainStack.push(x);

if (mainStack.size() == 1)

{

trackStack.push(x);

return;

}

if (x > trackStack.top())

trackStack.push(x);

else

trackStack.push(trackStack.top());

}

int getMax()

{

return trackStack.top();

}

int pop()

{

mainStack.pop();

trackStack.pop();

}

};

int main()

{

StackWithMax s;

s.push(20);

cout << s.getMax() << endl;

s.push(10);

cout << s.getMax() << endl;

s.push(50);

cout << s.getMax() << endl;

return 0;

}

Time Complexity : O(1)

**3. C++ program to implement a stack that supports get minimum ()**

#include <bits/stdc++.h>

using namespace std;

struct MyStack

{

stack<int> s;

int minEle;

void getMin()

{

if (s.empty())

cout << "Stack is empty\n";

else

cout <<"Minimum Element in the stack is: "

<< minEle << "\n";

}

void peek()

{

if (s.empty())

{

cout << "Stack is empty ";

return;

}

int t = s.top(); // Top element.

cout << "Top Most Element is: ";

(t < minEle)? cout << minEle: cout << t;

}

void pop()

{

if (s.empty())

{

cout << "Stack is empty\n";

return;

}

cout << "Top Most Element Removed: ";

int t = s.top();

s.pop();

if (t < minEle)

{

cout << minEle << "\n";

minEle = 2\*minEle - t;

}

else

cout << t << "\n";

}

void push(int x)

{ if (s.empty())

{

minEle = x;

s.push(x);

cout << "Number Inserted: " << x << "\n";

return;

}

if (x < minEle)

{

s.push(2\*x - minEle);

minEle = x;

}

else

s.push(x);

cout << "Number Inserted: " << x << "\n";

}

};

int main()

{

MyStack s;

s.push(3);

s.push(5);

s.getMin();

s.push(2);

s.push(1);

s.getMin();

s.pop();

s.getMin();

s.pop();

s.peek();

return 0;

}