**ASSIGNMENT 5:**

1.Write function for insertion sort

Sol:

#include <math.h>

#include <stdio.h>

void insertionSort(int arr[], int n)

{

int i, key, j;

for (i = 1; i < n; i++) {

key = arr[i];

j = i - 1;

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

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

void printArray(int arr[], int n)

{

int i;

for (i = 0; i < n; i++)

printf("%d ", arr[i]);

printf("\n"); }

int main()

{

int arr[] = { 12, 11, 13, 5, 6 };

int n = sizeof(arr) / sizeof(arr[0]);

insertionSort(arr, n);

printArray(arr, n);

return 0;

}

2.Write function to find maximum element in stack

Sol:

#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(50);

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

s.push(30);

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

s.push(10);

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

return 0;

}

3.Write a function to find minimum element in stack

Sol:

#include <iostream>

#include <stack>

 class Stack

{

        std::stack<int> s;

   int min;

 public:

 void push(int x)

     {

        if (s.empty()) {

            s.push(x);

            min = x;

        }

        else if (x > min) {

            s.push(x);

        }

        else {

            s.push(2 \* x - min);

            min = x;

        }

    }

void pop()

    {

        if (s.empty()) {

            std::cout << "Stack underflow!!" << '\n';

        }

 int top = s.top();

        if (top < min)

            min = 2 \* min - top;

        s.pop();

    }

int minimum()

    {

        return min;

    }

};

|  |
| --- |
| int main()  {      Stack s;      s.push(6);      std::cout << s.minimum() << '\n';        s.push(7);      std::cout << s.minimum() << '\n';        s.push(5);      std::cout << s.minimum() << '\n';        s.push(3);      std::cout << s.minimum() << '\n';        s.pop();      std::cout << s.minimum() << '\n';        s.pop();      std::cout << s.minimum() << '\n';        return 0;  } |