```cpp
    //
    //  main.cpp
    //  DSA practice for exam
    //
    //  Created by Harsh Anand on 05/12/2023.
    //

/*

 singly list☑️ /doubly / circular done
 stack ( implementation using array ) pranthesis check/ bracket check☑️
 infix to postfix ☑️
 queue(enqueue dequeue)

 sorting{
 bubble☑️
 insertion☑️
 selection☑️
 merge sort☑️
 quick sorting*/

    //}
#include <iostream>
#include <stdio.h>


using namespace std;

static int SIZE=0;

struct node  {
    int data;
    struct node * next;
    node(int x){  // constructor
        this->data=x;
        this->next=NULL;
    }
}*head, *tail;


void pushback( int x){
    SIZE++;
    node * temp= new node(x);
    if (!head) {
        head=tail=temp;
    }else{
        tail->next=temp;
        tail=temp;
    }
}
/*---------*/
```

```
void print(node *temp){
     if (SIZE>0) {
          while (temp) {
               cout<< (char)temp->data<<" ";
               temp=temp->next;
          }
          cout<<"\n";
     }else{
          cerr<<"invalid\n";
          return;
     }

}

/*---------*/

void create( int x){ // create linked list;
     while (x--) {
          int n;
          cin>>n;
          pushback(n);// calling push back fxn repetedly
     }
}
/*---------*/

void pushfront(int x){
     SIZE++;
     node *temp= new node(x);
     if (!head) {
          head=tail=temp;
     }else{
          temp->next=head;
          head=temp;
     }
}
/*---------*/

void popback(void){ //O(n) time taken as i have to triverse through whole
 linked list;
     if (SIZE<1) {
          cerr<<"no head\n"; // error
          return;

     }else if( SIZE ==1){
          free(head);
          SIZE--;
     }else{
          node * temp= head;
          for (int i=0; i<SIZE-2; ++i) {
               temp=temp->next;
          }
          node *x= temp->next;
          tail=temp;
          tail->next=NULL;
```

```
            free(x);  // not compalsary(but abhilash sir check kare tho
             compalsary  actually compilar automatically free the unused heap
             memory  ( memory management );
            SIZE--;
        }
}
/*---------*/
void popfront(void){ //O(n) time taken as i have to triverse through whole
 linked list;
        if (SIZE<1) {
            cerr<<"no head\n"; // error
            return;
        }else if( SIZE ==1){
            SIZE--;
            free(head);
        }else{
            SIZE--;
            node * x= head;
            head=head->next;
            free(x);
        }
}
/*---------*/

void insert(int x, int index){  // insert at index  // O(n)
        if (!head or index > SIZE or index<0) {
            cerr<<"invalid index\n";
        }else if(index==0){
            pushfront(x);
        }else if (index == SIZE){
            pushback(x);
        }else{
            SIZE++;
            node * temp= new node(x);
            node * x=head;
            for (int i=0; i<index-1; ++i) {
                x=x->next;
            }
            temp->next=x->next;
            x->next=temp;
        }
}
/*---------*/
void del(int index){  // delete at index
        if (!head or index > SIZE or index<0) {
            cerr<<"invalid index\n";
        }else if(index==0){
            popfront();
        }else if (index== SIZE){
            popback();
        }else{
            SIZE--;
            node * temp=nullptr;
            node * x=head;
            for (int i=0; i<index-1; ++i) {
                temp=x;
                x=x->next;
            }
            temp->next=x->next;
            free(x);
        }
}
/*---------*/
        // check pranthesis //

int bracket(char c){
        if (c=='(') return 1;//0
        if (c==')') return 2;//1
        if (c=='{') return 3;//2
        if (c=='}') return 4;//3
        if (c=='[') return 5;//4
        if (c==']') return 6;//5
        else return 0;
}
void pranthesis( string s){
        int ans[100];
        ans[0]=bracket(s[0]);
        int j=1;
        for (int i=1; s[i]; ++i) {
            if (!bracket(s[i])) {
                continue;
            }
            if (ans[j-1] - bracket(s[i])==-1) {
                j--;
            }else{
                ans[j++]= bracket(s[i]);
            }
        }
            // if j==0 balanced else not
        printf(j?"fuck no\n":"balanced\n");
}
/*-------------*/


        // infix to postfix

int Precedence(char c){
        if (c=='^') return 5;
        if (c=='/') return 4;
        if (c=='*') return 4;
        if (c=='-') return 3;
        if (c=='+') return 3;
        return 0;
}
void infix_to_postfix(string s){
        char stack_m[1000];
        int index=-1 ;
        for (int i=0; s[i] ; ++i) {
            if (s[i]=='(') {
                stack_m[++index]='(';
          }else if (s[i]==')'){
               while (stack_m[index]!='(')  cout<<stack_m[index--];
               index--;
          }else if (Precedence(s[i])!=0){
               while (index!=-1 and
                !(Precedence(s[i])>Precedence(stack_m[index]))) {
                    cout<<stack_m[index];
                    index--;
               }
               stack_m[++index]=s[i];
          }else cout<<s[i];
     }
     while (index!=-1) {
          cout<<stack_m[index--];
     }
}
/*------------*/
     //sorting
void swap(int * a, int *b){
     int c=*a;
     *a=*b;
     *b=c;
}

     //bubble sort//
void bubble(int *a, int n){
     for (int i=0; i<n; ++i) {
          for (int j=0; j<n; ++j) {
               if (a[i]<a[j]) {
                    swap(&a[i], &a[j]);
               }
          }
     }
}

     // insertion sort
void insertion_sort(int *arr, int n){
     int i=0,j=0,k;
     for (; i<n; ++i) {
          j=i-1;
          k=arr[i];
          while (j>-1 and arr[j]>k) {
               arr[j+1]=arr[j];
               j--;
          }
          arr[j+1]=k;
     }

}


/* merge sort*/
void kajukatli(int *arr, int low, int mid, int high){
     int res[100];
     int i=low,j=mid+1, k=low;
```

```
    while (i<=mid and j<=high) {
        if (arr[i]<arr[j]) {
            res[k++]=arr[i++];
        }else{
            res[k++]=arr[j++];
        }
    }
    for (; i<=mid; ++i) {
        res[k++]=arr[i];
    }
    for (; j<=high; ++j) {
        res[k++]=arr[j];
    }
    for (int i=low; i<=high; ++i) {
        arr[i]=res[i];
    }
}

void merge_sort(int * arr, int l, int h){
    if (l<h) {
        int mid=(l+h)/2;
        merge_sort(arr, l, mid);
        merge_sort(arr, mid+1,h);
        kajukatli(arr, l, mid, h);
    }

}
/*----*/

/*selectio sort*/

void selection(int *a, int n){
    int i,j,k;
    for (i=0; i<n-1; ++i) {
        for (j=k=i; j<n; ++j) {
            if (a[j]<a[k]) {
                k=j;
            }
        }
        swap(a[i], a[k]);
    }
}

/*---*/



int main(int argc, const char * argv[]) {
    freopen("/Users/harshanand/Desktop/C++ file/DSA practice for
     exam/input.txt", "r", stdin);
    freopen("/Users/harshanand/Desktop/C++ file/DSA practice for
     exam/output.txt", "w", stdout);
    freopen("/Users/harshanand/Desktop/C++ file/DSA practice for
     exam/error.txt", "w", stderr);
    int n;
    cin>>n;
    int arr[n+1];
    int i=0;
    for(; i<n; ++i){
        cin>>arr[i];
    }

    arr[i]=INT_MAX;
        //    merge_sort(arr, 0, n);
        //     selection(arr, n);
    insertion_sort(arr, n);
    for(int i=0; i<n; ++i){
        cout<<arr[i]<<" ";
    }

    return 0;
}
```