**Name :**

**CS 2ND YEAR**

**SECTION :**

**ROLL NO.:**

**ENROLLMENT NO.:**

**SUBJECT : DATA STRUCTURE AND ALGORITHM LAB**

**[WEEK : 5]**

**ASSIGNMENT : 5**

**HACKERRANK ID :**

**DATE : 04.08.2020**

**Q1. You have set of numbers stores in a linked list. Take another number n as user input and insert the element in a position in the linked list. the position of insetion is also given by the user.**

**Code :**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*CreateList(struct node \*head);

void DisplayList(struct node \*head);

struct node \*InsertList(struct node \*head,int n,int p);

int main()

{

int x,n,p,i=0;

struct node \*head=NULL;

scanf("%d %d %d",&x,&n,&p);

while(i<x)

{

head=CreateList(head);

i++;

}

head=InsertList(head,n,p);

DisplayList(head);

return 0;

}

struct node \*CreateList(struct node \*head)

{

int a;

struct node \*ptr,\*newnode=(struct node\*)malloc(sizeof(struct node));

if(newnode==NULL)

{

printf("Memory insertion not possible.");

exit(0);

}

scanf("%d",&a);

newnode->data=a;

newnode->next=NULL;

if(head==NULL)

{

head=newnode;

}

else

{

ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=newnode;

}

return head;

}

void DisplayList(struct node \*head)

{

struct node \*ptr;

if(head==NULL)

{

printf("The list is empty.");

}

else

{

ptr=head;

while(ptr!=NULL)

{

printf("%d ",ptr->data);

ptr=ptr->next;

}

}

}

struct node \*InsertList(struct node \*head,int n,int p)

{

int i=1;

struct node \*ptr=head,\*ptr1;

while(ptr!=NULL&&i<p)

{

ptr1=ptr;

ptr=ptr->next;

i++;

}

struct node \*newnode=(struct node\*)malloc(sizeof(struct node));

if(newnode==NULL)

{

printf("Memory underflow. No insertion possible.");

exit(0);

}

newnode->data=n;

newnode->next=NULL;

newnode->next=ptr1->next;

ptr1->next=newnode;

return head;

}

**Q2. Take a linked list as user input. Copy all the unique elements from that linked list into another new linked list.**

**Code :**

#include <stdio.h>

#define MAX\_SIZE 100 // Maximum size of the array

int main()

{

int arr[MAX\_SIZE]; // Declares an array of size 100

int size; // Total number of elements in array

int i, j, k; // Loop control variables

/\* Input size of the array \*/

scanf("%d", &size);

/\* Input elements in the array \*/

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

{

scanf("%d", &arr[i]);

}

/\*

\* Find duplicate elements in array

\*/

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

{

for(j=i+1; j<size; j++)

{

/\* If any duplicate found \*/

if(arr[i] == arr[j])

{

/\* Delete the current duplicate element \*/

for(k=j; k<size; k++)

{

arr[k] = arr[k + 1];

}

/\* Decrement size after removing duplicate element \*/

size--;

/\* If shifting of elements occur then don't increment j \*/

j--;

}

}

}

/\*

\* Print array after deleting duplicate elements

\*/

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

{

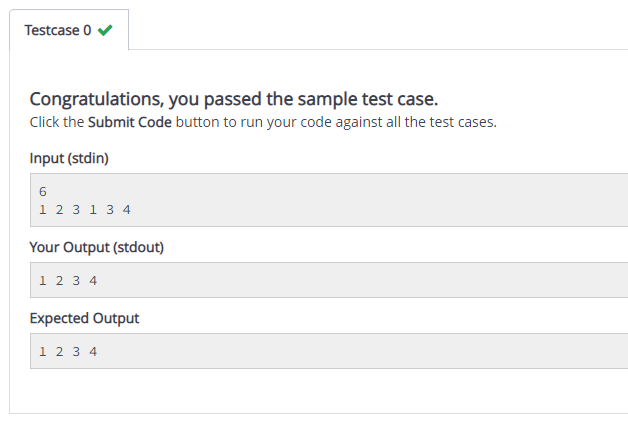
printf("%d ", arr[i]);

}

return 0;

}

**Output :**

****

**Q3. Add two given polynomil using linked list.**

**Code :**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int coeff;

int exp;

struct node \*next;

};

struct node \*CreateList(struct node \*head,int n);

void DisplayList(struct node \*head);

struct node \*PolyAdd(struct node \*head1,struct node \*head2,struct node \*head3);

struct node \*AddPolyNode(struct node \*head,int c,int e);

int main()

{

struct node \*head1=NULL,\*head2=NULL,\*head3=NULL;

int n1,n2;

scanf("%d %d",&n1,&n2);

while(n1>=0)

{

head1=CreateList(head1,n1);

n1--;

}

while(n2>=0)

{

head2=CreateList(head2,n2);

n2--;

}

head3=PolyAdd(head1,head2,head3);

DisplayList(head3);

return 0;

}

struct node \*CreateList(struct node \*head,int n)

{

struct node \*newnode=(struct node\*)malloc(sizeof(struct node));

if(newnode==NULL)

{

printf("Memory underflow. No insertion possible.");

exit(0);

}

int a;

scanf("%d",&a);

newnode->coeff=a;

newnode->exp=n;

newnode->next=NULL;

if(head==NULL)

{

head=newnode;

}

else

{

struct node \*ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=newnode;

}

return head;

}

void DisplayList(struct node \*head)

{

if(head==NULL)

{

printf("The list is empty");

}

else

{

struct node \*ptr=head;

while(ptr!=NULL)

{

if(ptr->next==NULL)

{

printf("%d",ptr->coeff);

}

else if(ptr->exp==1)

{

printf("%dx + ",ptr->coeff);

}

else

{

printf("%dx^%d + ",ptr->coeff,ptr->exp);

}

ptr=ptr->next;

}

}

}

struct node \*AddPolyNode(struct node \*head,int c,int e)

{

struct node \*newnode=(struct node\*)malloc(sizeof(struct node));

if(newnode==NULL)

{

printf("Memory underflow. No insertion possible.");

exit(0);

}

newnode->coeff=c;

newnode->exp=e;

newnode->next=NULL;

if(head==NULL)

{

head=newnode;

}

else

{

struct node \*ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=newnode;

}

return head;

}

struct node \*PolyAdd(struct node \*head1,struct node \*head2,struct node \*head3)

{

struct node \*ptr1=head1,\*ptr2=head2;

while(ptr1!=NULL&&ptr2!=NULL)

{

if(ptr1->exp==ptr2->exp)

{

head3=AddPolyNode(head3,ptr1->coeff+ptr2->coeff,ptr1->exp);

ptr1=ptr1->next;

ptr2=ptr2->next;

}

else if(ptr1->exp>ptr2->exp)

{

head3=AddPolyNode(head3,ptr1->coeff,ptr1->exp);

ptr1=ptr1->next;

}

else

{

head3=AddPolyNode(head3,ptr2->coeff,ptr2->exp);

ptr2=ptr2->next;

}

}

if(ptr1!=NULL)

{

while(ptr1!=NULL)

{

head3=AddPolyNode(head3,ptr1->coeff,ptr1->exp);

ptr1=ptr1->next;

}

}

if(ptr2!=NULL)

{

while(ptr2!=NULL)

{

head3=AddPolyNode(head3,ptr2->coeff,ptr2->exp);

ptr2=ptr2->next;

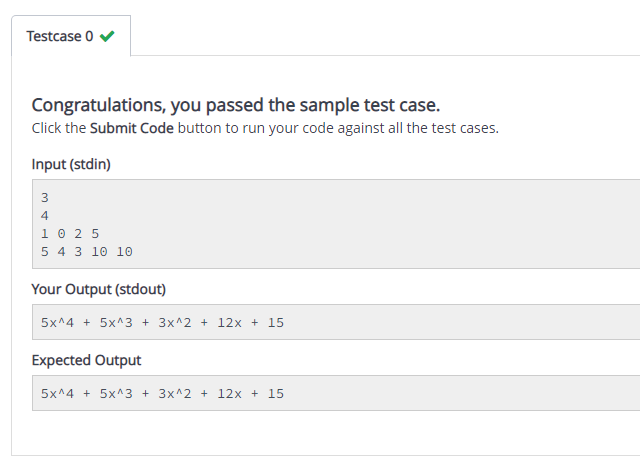
}

}

return head3;

}

**Output :**

****

**Q4. Multiply two given polynomial using linked list.**

**Code :**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int coeff;

int exp;

struct node \*next;

};

struct node \*CreateList(struct node \*head,int e);

void DisplayList(struct node \*head);

struct node \*MulPoly(struct node \*head1,struct node \*head2,struct node \*head3);

struct node \*CreateNodeMul(struct node \*head,int c,int i);

struct node \*CommonReduction(struct node \*head);

int main()

{

struct node \*head1=NULL,\*head2=NULL,\*head3=NULL;

int n1,n2;

scanf("%d %d",&n1,&n2);

while(n1>=0)

{

head1=CreateList(head1,n1);

n1--;

}

while(n2>=0)

{

head2=CreateList(head2,n2);

n2--;

}

head3=MulPoly(head1,head2,head3);

DisplayList(head3);

return 0;

}

struct node \*CreateList(struct node \*head,int e)

{

struct node newnode=(struct node)malloc(sizeof(struct node));

if(newnode==NULL)

{

printf("Memory underflow. No insertion possible.");

exit(0);

}

int a;

scanf("%d",&a);

newnode->coeff=a;

newnode->exp=e;

newnode->next=NULL;

if(head==NULL)

{

head=newnode;

}

else

{

struct node \*ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=newnode;

}

return head;

}

void DisplayList(struct node \*head)

{

if(head==NULL)

{

printf("The list is empty.");

}

else

{

struct node \*ptr=head;

while(ptr!=NULL)

{

if(ptr->next==NULL)

{

printf("%d",ptr->coeff);

}

else if(ptr->exp==1)

{

printf("%dx + ",ptr->coeff);

}

else

{

printf("%dx^%d + ",ptr->coeff,ptr->exp);

}

ptr=ptr->next;

}

}

}

struct node \*MulPoly(struct node \*head1,struct node \*head2,struct node \*head3)

{

int c,e;

struct node \*ptr1=head1,\*ptr2;

while(ptr1!=NULL)

{

ptr2=head2;

while(ptr2!=NULL)

{

c=(ptr1->coeff)\*(ptr2->coeff);

e=(ptr1->exp)+(ptr2->exp);

head3=CreateNodeMul(head3,c,e);

ptr2=ptr2->next;

}

ptr1=ptr1->next;

}

head3=CommonReduction(head3);

return head3;

}

struct node \*CreateNodeMul(struct node \*head,int c,int e)

{

struct node newnode=(struct node)malloc(sizeof(struct node));

if(newnode==NULL)

{

printf("Memory underflow. No insertion possible.");

exit(0);

}

newnode->exp=e;

newnode->coeff=c;

newnode->next=NULL;

if(head==NULL)

{

head=newnode;

}

else

{

struct node \*ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=newnode;

}

return head;

}

struct node \*CommonReduction(struct node \*head)

{

struct node \*ptr1=head,\*ptr2,\*del;

while(ptr1->next!=NULL)

{

ptr2=ptr1;

while(ptr2->next!=NULL)

{

if(ptr1->exp==ptr2->next->exp)

{

ptr1->coeff+=ptr2->next->coeff;

del=ptr2->next;

ptr2->next=ptr2->next->next;

del->next=NULL;

free(del);

}

else

{

ptr2=ptr2->next;

}

}

ptr1=ptr1->next;

}

return head;

}

#include<stdio.h>

#include<stdlib.h>

struct node

{

int coeff;

int exp;

struct node \*next;

};

struct node \*CreateList(struct node \*head,int e);

void DisplayList(struct node \*head);

struct node \*MulPoly(struct node \*head1,struct node \*head2,struct node \*head3);

struct node \*CreateNodeMul(struct node \*head,int c,int i);

struct node \*CommonReduction(struct node \*head);

int main()

{

struct node \*head1=NULL,\*head2=NULL,\*head3=NULL;

int n1,n2;

scanf("%d %d",&n1,&n2);

while(n1>=0)

{

head1=CreateList(head1,n1);

n1--;

}

while(n2>=0)

{

head2=CreateList(head2,n2);

n2--;

}

head3=MulPoly(head1,head2,head3);

DisplayList(head3);

return 0;

}

struct node \*CreateList(struct node \*head,int e)

{

struct node newnode=(struct node)malloc(sizeof(struct node));

if(newnode==NULL)

{

printf("Memory underflow. No insertion possible.");

exit(0);

}

int a;

scanf("%d",&a);

newnode->coeff=a;

newnode->exp=e;

newnode->next=NULL;

if(head==NULL)

{

head=newnode;

}

else

{

struct node \*ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=newnode;

}

return head;

}

void DisplayList(struct node \*head)

{

if(head==NULL)

{

printf("The list is empty.");

}

else

{

struct node \*ptr=head;

while(ptr!=NULL)

{

if(ptr->next==NULL)

{

printf("%d",ptr->coeff);

}

else if(ptr->exp==1)

{

printf("%dx + ",ptr->coeff);

}

else

{

printf("%dx^%d + ",ptr->coeff,ptr->exp);

}

ptr=ptr->next;

}

}

}

struct node \*MulPoly(struct node \*head1,struct node \*head2,struct node \*head3)

{

int c,e;

struct node \*ptr1=head1,\*ptr2;

while(ptr1!=NULL)

{

ptr2=head2;

while(ptr2!=NULL)

{

c=(ptr1->coeff)\*(ptr2->coeff);

e=(ptr1->exp)+(ptr2->exp);

head3=CreateNodeMul(head3,c,e);

ptr2=ptr2->next;

}

ptr1=ptr1->next;

}

head3=CommonReduction(head3);

return head3;

}

struct node \*CreateNodeMul(struct node \*head,int c,int e)

{

struct node newnode=(struct node)malloc(sizeof(struct node));

if(newnode==NULL)

{

printf("Memory underflow. No insertion possible.");

exit(0);

}

newnode->exp=e;

newnode->coeff=c;

newnode->next=NULL;

if(head==NULL)

{

head=newnode;

}

else

{

struct node \*ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=newnode;

}

return head;

}

struct node \*CommonReduction(struct node \*head)

{

struct node \*ptr1=head,\*ptr2,\*del;

while(ptr1->next!=NULL)

{

ptr2=ptr1;

while(ptr2->next!=NULL)

{

if(ptr1->exp==ptr2->next->exp)

{

ptr1->coeff+=ptr2->next->coeff;

del=ptr2->next;

ptr2->next=ptr2->next->next;

del->next=NULL;

free(del);

}

else

{

ptr2=ptr2->next;

}

}

ptr1=ptr1->next;

}

return head;

}

**Q5. You have a set of numbers stored in a linked list. Find the sum of all numbers in even positions in a linked list**

**Code :**

#include <stdio.h>

#include <stdlib.h>

/\* Link list node \*/

struct Node {

int data;

struct Node\* next;

};

/\* Function to get the alternate

nodes of the linked list \*/

int sumAlternateNode(struct Node\* head)

{

int count = 0;

int sum = 0;

while (head != NULL) {

// when count is even sum the nodes

if (count % 2 == 0)

sum += head->data;

// count the nodes

count++;

// move on the next node.

head = head->next;

}

return sum;

}

// Function to push node at head

void push(struct Node\*\* head\_ref, int new\_data)

{

struct Node\* new\_node = (struct Node\*)malloc(sizeof(struct Node));

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

// Driver code

int main()

{

/\* Start with the empty list \*/

struct Node\* head = NULL;

/\* Use push() function to construct

the below list 1 -> 5 -> 2 -> 4 -> 8 -> 6 \*/

push(&head, 1);

push(&head, 5);

push(&head, 2);

push(&head, 4);

push(&head, 8);

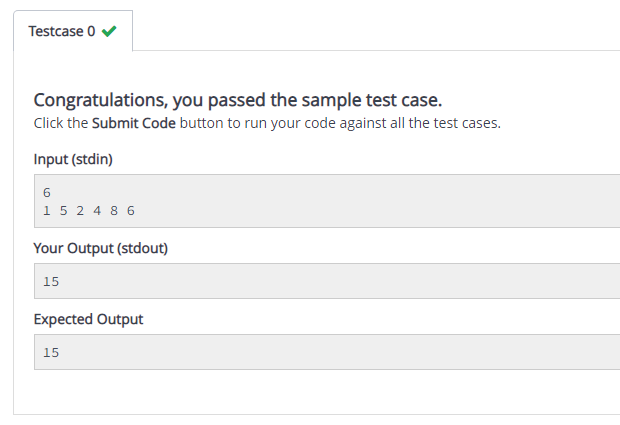
push(&head, 6);

printf("%d", sumAlternateNode(head));

return 0;

}

**Output :**

****

**Q6. You have a set of elements stored in a linked lsit. Delete all the odd numbers from the linked list.**

**Code :**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*prev;

struct node \*next;

};

struct node \*CreateList(struct node \*head);

void DisplayList(struct node \*head);

struct node \*DeleteOdd(struct node \*head);

int main()

{

int n,i=1;

struct node \*head=NULL;

scanf("%d",&n);

while(i<=n)

{

head=CreateList(head);

i++;

}

head=DeleteOdd(head);

DisplayList(head);

return 0;

}

struct node \*CreateList(struct node \*head)

{

struct node \*newnode=(struct node\*)malloc(sizeof(struct node));

int i;

if(newnode==NULL)

{

printf("Memory underflow. Insertion not possible.");

exit(0);

}

scanf("%d",&i);

newnode->data=i;

newnode->next=NULL;

newnode->prev=NULL;

if(head==NULL)

{

head=newnode;

}

else

{

struct node \*ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=newnode;

newnode->prev=ptr;

}

return head;

}

struct node \*DeleteOdd(struct node \*head)

{

struct node \*ptr=head,\*del,\*q,\*p;

while(ptr!=NULL)

{

if(ptr->data%2!=0)

{

if(ptr==head)

{

del=ptr;

ptr=ptr->next;

ptr->prev=NULL;

head=ptr;

del->next=NULL;

free(del);

}

else

{

del=ptr;

ptr=ptr->next;

q=del->prev;

p=del->next;

q->next=p;

if(p!=NULL)

{

p->prev=q;

}

del->next=del->prev=NULL;

free(del);

}

}

else

{

ptr=ptr->next;

}

}

return head;

}

void DisplayList(struct node \*head)

{

if(head==NULL)

{

printf("The list is empty.");

}

else

{

struct node \*ptr=head;

while(ptr!=NULL)

{

printf("%d ",ptr->data);

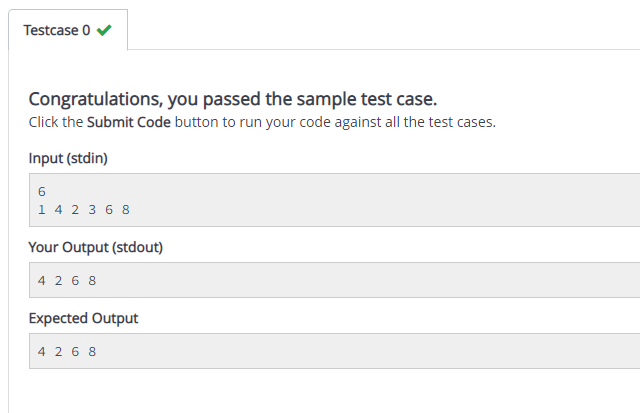
ptr=ptr->next;

}

}

}

**Output :**

****

**Q7. You have two linked lists given. Merge them into a single list.**

**Code :**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*CreateList(struct node \*head);

struct node \*CreateMergeList(struct node \*head,int n);

void DisplayList(struct node \*head);

struct node \*MergeList(struct node \*head1,struct node \*head2,struct node \*head3);

int main()

{

struct node \*head1=NULL,\*head2=NULL,\*head3=NULL;

int n1,n2,i;

i=1;

scanf("%d %d",&n1,&n2);

while(i<=n1)

{

head1=CreateList(head1);

i++;

}

i=1;

while(i<=n2)

{

head2=CreateList(head2);

i++;

}

head3=MergeList(head1,head2,head3);

DisplayList(head3);

return 0;

}

struct node \*CreateList(struct node \*head)

{

struct node \*newnode=(struct node\*)malloc(sizeof(struct node));

int i;

if(newnode==NULL)

{

printf("Memory underflow. Insertion not possible.");

exit(0);

}

scanf("%d",&i);

newnode->data=i;

newnode->next=NULL;

if(head==NULL)

{

head=newnode;

}

else

{

struct node \*ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=newnode;

}

return head;

}

void DisplayList(struct node \*head)

{

if(head==NULL)

{

printf("The list is empty.");

}

else

{

struct node \*ptr=head;

while(ptr!=NULL)

{

printf("%d ",ptr->data);

ptr=ptr->next;

}

}

}

struct node \*CreateMergeList(struct node \*head,int n)

{

struct node \*newnode=(struct node\*)malloc(sizeof(struct node));

if(newnode==NULL)

{

printf("Memory underflow. Insertion not possible.");

exit(0);

}

newnode->data=n;

newnode->next=NULL;

if(head==NULL)

{

head=newnode;

}

else

{

struct node \*ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=newnode;

}

return head;

}

struct node \*MergeList(struct node \*head1,struct node \*head2,struct node \*head3)

{

struct node \*ptr1=head1,\*ptr2=head2;

while(ptr1!=NULL&&ptr2!=NULL)

{

if(ptr1->data<=ptr2->data)

{

head3=CreateMergeList(head3,ptr1->data);

ptr1=ptr1->next;

}

else if(ptr1->data>ptr2->data)

{

head3=CreateMergeList(head3,ptr2->data);

ptr2=ptr2->next;

}

}

if(ptr1!=NULL)

{

while(ptr1!=NULL)

{

head3=CreateMergeList(head3,ptr1->data);

ptr1=ptr1->next;

}

}

if(ptr2!=NULL)

{

while(ptr2!=NULL)

{

head3=CreateMergeList(head3,ptr2->data);

ptr2=ptr2->next;

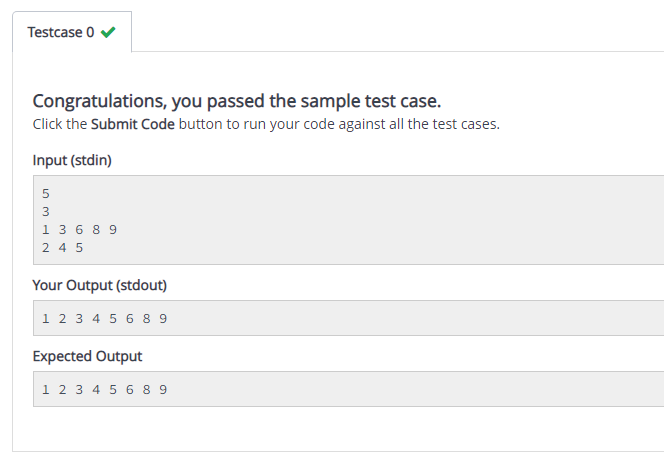
}

}

return head3;

}

**Output :**

****

**Q8. You have a set of n numbers stored in a linked list. Sort the list using bubble sort**

**Code :**

#include <stdio.h>

//Represent a node of the singly linked list

struct node{

int data;

struct node \*next;

};

//Represent the head and tail of the singly linked list

struct node \*head, \*tail = NULL;

//addNode() will add a new node to the list

void addNode(int data) {

//Create a new node

struct node \*newNode = (struct node\*)malloc(sizeof(struct node));

newNode->data = data;

newNode->next = NULL;

//Checks if the list is empty

if(head == NULL) {

//If list is empty, both head and tail will point to new node

head = newNode;

tail = newNode;

}

else {

//newNode will be added after tail such that tail's next will point to newNode

tail->next = newNode;

//newNode will become new tail of the list

tail = newNode;

}

}

//sortList() will sort nodes of the list in ascending order

void sortList() {

//Node current will point to head

struct node \*current = head, \*index = NULL;

int temp;

if(head == NULL) {

return;

}

else {

while(current != NULL) {

//Node index will point to node next to current

index = current->next;

while(index != NULL) {

//If current node's data is greater than index's node data, swap the data between them

if(current->data > index->data) {

temp = current->data;

current->data = index->data;

index->data = temp;

}

index = index->next;

}

current = current->next;

}

}

}

//display() will display all the nodes present in the list

void display() {

//Node current will point to head

struct node \*current = head;

if(head == NULL) {

printf("List is empty \n");

return;

}

while(current != NULL) {

//Prints each node by incrementing pointer

printf("%d ", current->data);

current = current->next;

}

printf("\n");

}

int main()

{

int n,data;

scanf("%d",&n);

//Adds data to the list

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

{

scanf("%d",&data);

addNode(data);

}

//Displaying original list

//Sorting list

sortList();

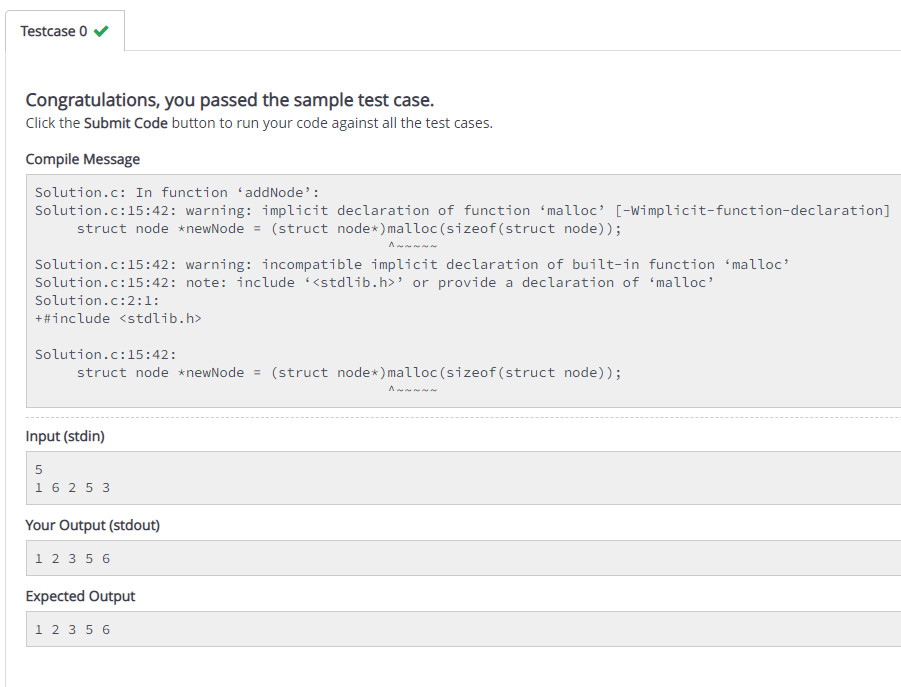
//Displaying sorted list

display();

return 0;

}

**Output :**

****