Rajalakshmi Engineering College

Name: Manisha S
Email:
240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE
AF Batch: 2028
Degree: B.E - ECE

Scan to verify

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 1

Attempt : 1
Total Mark :
10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Janani is a tech enthusiast who loves working with polynomials. She wants to create a program that can add polynomial coefficients and provide the sum of their coefficients.

The polynomials will be represented as a linked list, where each node of the linked list contains a coefficient and an exponent. The polynomial is represented in the standard form with descending order of exponents.

Input Format

The first line of input consists of an integer n, representing the number of terms in the first polynomial.

The following n lines of input consist of two integers each: the coefficient and the exponent of the term in the first polynomial.

The next line of input consists of an integer m, representing the number of terms in the second polynomial.

The following m lines of input consist of two integers each: the coefficient and the exponent of the term in the second polynomial.

Output Format

The output prints the sum of the coefficients of the polynomials.

Sample Test Case

```
Input: 3
2 2
3 1
4 0
3
2 2
3 1
4 0
Output: 18
```

Answer

```
// You are using
GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct poly
{
   int
   coeff;
   int
   expon;
   struct poly* next;
}Node;
Node* newnode(int coeff, int expon){
   Node* new_node = (Node*)
   malloc(sizeof(Node)); new_node->coeff =
   coeff;
   new_node->expon =
   expon; new_node->next =
   NULL; return new_node;
}
void insertNode(Node** head, int coeff, int
   expon){ Node* temp = *head;
   if(temp == NULL){
      *head =
      newnode(coeff,expon);
```

```
    }
    while(temp->next !=
       NULL){ temp =
       temp->next;
    }
    temp->next =
    newnode(coeff,expon);
}
int main()
{
    int n,coeff,expon;
    scanf("%d",&n);
    Node* poly1;
    Node* poly2;
    for(int i=0; i<n;
    i++)
    {
       scanf("%d %d",&coeff,&expon);
       insertNode(&poly1, coeff,
       expon);
    }
    scanf("%d",&n);
    for(int i=0; i<n;
    i++)
    {
       scanf("%d %d",&coeff,
       &expon); insertNode(&poly2,
       coeff, expon);
    }
    int sum = 0;
    while(poly1 !=
    NULL)
    {
       sum +=
       poly1->coeff; poly1
       = poly1->next;
    }
    while(poly2 != NULL)
    {
       sum +=
       poly2->coeff; poly2
       = poly2->next;
    }
    printf("%d",sum);
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE
AF Batch: 2028
Degree: B.E - ECE

Scan to verify

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Arun is learning about data structures and algorithms. He needs your help in solving a specific problem related to a singly linked list.

Your task is to implement a program to delete a node at a given position. If the position is valid, the program should perform the deletion; otherwise, it should display an appropriate message.

Input Format

The first line of input consists of an integer N, representing the number of elements in the linked list.

The second line consists of N space-separated elements of the

linked list. The third line consists of an integer x, representing the

position to delete.

Position starts from 1.

Output Format

The output prints space-separated integers, representing the updated linked list after deleting the element at the given position.

If the position is not valid, print "Invalid position. Deletion not possible."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
8 2 3 1 7
2
Output: 8 3 1 7

Answer

```
#include
<stdio.h>
#include
<stdlib.h>

void insert(int);
void
display_List();
void deleteNode(int);

struct node
   { int
   data;
   struct node* next;
} *head = NULL, *tail = NULL;

void insert(int value)
{
   struct node* newNode=(struct node*)malloc(sizeof(struct
   node)); newNode->data=value;
   newNode->next=NULL;

   if(head==NULL)
   {
      head=newNode;
      tail=newNode;
```

```
    }
    else{
       tail->next=new
       Node;
       tail=newNode;
    }
 }
 void deleteNode(int position)
 {
    if(head==NULL)
    {
       printf("Invalid position.DE11etion not
       possible.\n"); return;
    }
    struct node* temp=head;

    if(position ==1)
    {
       head
       =temp->n
       ext; free(temp);
       display_List();
       return;
    }
    for(int i=1;temp != NULL && i<position - 1;i++)
    {
       temp=temp->next;
    }
    if(temp==NULL || temp->next==NULL)
    {
       printf("Invalid position.  Deletion not
       possible.\n"); return;
    }
    struct node* nodeToDelete
    =temp->next; temp->next =
    temp->next->next; free(nodeToDelete);

    display_List();
 }
 void display_List()
 {
    if(head==NULL){
        return;
    }
    struct node*
    temp=head;
    while(temp!= NULL)
    {
       printf
       ("%d",temp-
       >data);
       if(temp->next!= NULL)
       {
          printf(" ");
       }
       temp=temp->next;
    }
    printf("\n");
}
int main() {
    int num_elements, element,
    pos_to_delete;

    scanf("%d", &num_elements);

    for (int i = 0; i < num_elements;
       i++) { scanf("%d", &element);
       insert(element);
    }
    scanf("%d",

    &pos_to_delete);

    deleteNode(pos_to_delete);

    return 0;
}
```

Status : Correct *Marks :* *10/10*

Rajalakshmi Engineering College

Name: Manisha S
Email:
240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE
AF Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Imagine you are working on a text processing tool and need to implement a feature that allows users to insert characters at a specific position.

Implement a program that takes user inputs to create a singly linked list of characters and inserts a new character after a given index in the list.

Input Format

The first line of input consists of an integer N, representing the number of characters in the linked list.

The second line consists of a sequence of N characters, representing the linked list.

The third line consists of an integer index, representing the index(0-based) after which the new character node needs to be inserted.

The fourth line consists of a character value representing the character to be inserted after the given index.

Output Format

If the provided index is out of bounds (larger than the list size):

1. The first line of output prints "Invalid index".
2. The second line prints "Updated list: " followed by the unchanged linked list values.

Otherwise, the output prints "Updated list: " followed by the updated linked list after inserting the new character after the given index.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
a b c d
e 2
X

Output: Updated list: a b c X d e

Answer

```
#include<stdio.h>
#include<stdlib.h>
typedef struct
Char {
   char value;
   struct Char* next;
}Node;
Node*newnode(char value) {
   Node* new_node = (Node*)
   malloc(sizeof(Node)); new_node->value =
   value;
   new_node->next =
   NULL; return new_node;
}
```

```
void insertNode(Node** head, char
   value){ Node* temp = *head;
   if(temp == NULL) {
      *head =
      newnode(value);
      return;
   }
   while(temp->next != NULL)
      { temp = temp->next;
   }
   temp->next = newnode(value);
}
int length(Node*
   head) { int len = 0;
   while(head != NULL)
      { head
      =head->next;
      len++;
   }
   return len;
}
void traverse(Node*
   head) { while(head !=
   NULL) {
      printf(
      "%c",head->v
      alue); head =
      head->next;
   }
   printf("\n");
}
void insert(Node** head, int pos, char
   value){ if(pos >= length(*head)) {
      printf("Invalid
      index\n"); return;
   }
   Node* temp = *head;
   for(int i=0; i<pos; i++)
   {
      temp = temp->next;
   }
   Node* new_node =
   newnode(value); new_node->next
   = temp->next;  temp->next =
   new_node;
}
int main()
```

```
    char value;
    Node* head = NULL;
    scanf("%d",&n);
    for(int i=0; i<=n; i++)
    {
       scanf("%c ",&value);
       if(value == ' ' || value ==
          '\n') { continue;
       }
       insertNode(&head, value);
    }
    scanf("%d %c"
    ,&n,&value);
    insert(&head, n,
    value); printf("Updated
    list: "); traverse(head);
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE
AF Batch: 2028
Degree: B.E - ECE

Scan to verify

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

As part of a programming assignment in a data structures course, students are required to create a program to construct a singly linked list by inserting elements at the beginning.

You are an evaluator of the course and guide the students to complete the task.

Input Format

The first line of input consists of an integer N, which is the number of elements.

The second line consists of N space-separated integers.

Output Format

The output prints the singly linked list elements, after inserting them at the beginning.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
78 89 34 51 67

Output: 67 51 34 89 78

Answer

```
#include
<stdio.h>
#include
<stdlib.h>

struct Node
   { int data;
   struct Node* next;
};

struct Node *head=NULL,*newnode,*ptr;
void insertAtFront(struct Node **head,int
a)
{newnode=(struct Node*)malloc(sizeof(struct
Node)); newnode->data=a;
newnode->next=NULL;
newnode->n
ext=*head;
   *head=newnode;

}
void printList(struct Node*head)
{ptr=head;
   while(ptr!=NULL
   )
   {printf("%d
",ptr->data);
   ptr=ptr->next;}
}

int main(){
   struct Node* head = NULL;
   for (int i = 0; i < n; i++)
      { int activity;
      scanf("%d",
      &activity);
      insertAtFront(&head,
      activity);
   }

   printList(head);
   struct Node* current =
   head; while (current !=
   NULL) {
      struct Node* temp =
      current; current =
      current->next; free(temp);
   }

   return 0;
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE
AF Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Imagine you are tasked with developing a simple GPA management system using a singly linked list. The system allows users to input student GPA values, insertion should happen at the front of the linked list, delete record by position, and display the updated list of student GPAs.

Input Format

The first line of input contains an integer n, representing the number of students.

The next n lines contain a single floating-point value representing the GPA of each student.

The last line contains an integer position, indicating the position at which a student record should be deleted. Position starts from 1.

Output Format

After deleting the data in the given position, display the output in the format "GPA: " followed by the GPA value, rounded off to one decimal place.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
3.8
3.2
3.5
4.1
2

Output: GPA: 4.1
GPA: 3.2
GPA: 3.8

Answer

```
#include<stdio.h>
#include<stdlib.h>
typedef     struct
gpa
{
   float value;
   struct gpa*
   next;
}Node;
Node* newnode(float value)
{
   Node* newgpa = (Node*)
   malloc(sizeof(Node)); newgpa->value =
   value;
   newgpa->next  =
   NULL; return
   newgpa;
}
Node* insertAtStart(Node* head, float value)
{
   Node* newgpa =
   newnode(value); newgpa->next
   = head;
   return newgpa;
}
```

```
void traverse(Node* head)
{
   while(head != NULL)
   {
      print
      f("
      GPA: %
      .1f\n",head->value
      ); head = head->next;
   }
}
void deleteAtPosition(Node** head, int
pos)
{
   pos -= 1;
   Node* temp =
   *head; if(pos == 0)
   {
      *head =
      temp->next;
      free(temp);
      return;
   }
   while(--pos)
   {
      temp = temp->next;
   }
   Node* temp1 = temp->next;
   temp->next =
   temp->next->next;
   free(temp1);
}
int main()
{
   int n,pos;
   float
   value;
   scanf("%d",&n);
   Node* head =
   NULL; for(int i=0;
   i<n; i++)
   {
      scanf("%f",&value);
      head = insertAtStart(head, value);
   }
   scanf("%d",&pos);
   deleteAtPosition(&head,
```

Status : Correct *Marks :* *10/10*

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE
AF Batch: 2028
Degree: B.E - ECE

Scan to verify

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 6

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John is tasked with creating a program to manage student roll numbers using a singly linked list.

Write a program for John that accepts students' roll numbers, inserts them at the end of the linked list, and displays the numbers.

Input Format

The first line of input consists of an integer N, representing the number of students.

The second line consists of N space-separated integers, representing the roll numbers of students.

Output Format

The output prints the space-separated integers singly linked list, after inserting the roll numbers of students at the end.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
23 85 47 62 31

Output: 23 85 47 62 31

Answer

```
#include<stdio.h>
#include<stdlib.h>
typedef struct
student
{
   int roll;
   struct student* next;
}Node;
Node* newnode(int rollno)
{
   Node* data = (Node*)
   malloc(sizeof(Node)); data->roll = rollno;
   data->next =
   NULL; return
   data;
}
void traverse(Node* head)
{
   while(head != NULL)
   {
      printf("%d
      ",head->roll); head =
      head->next;
   }
}
int main()
{
   int n,rollno;
   scanf("%d",&n);
   scanf("%d",&rollno)
   ;
   Node* head = newnode(rollno);
```

```
    Node* temp =
    head; while(--n)
    {
        scanf("%d",&rollno);
        temp->next =
        newnode(rollno); temp =
        temp->next;
    }
    traverse(head);
}
```

Status : Correct

Marks : *10/10*

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE
AF Batch: 2028
Degree: B.E - ECE

Scan to verify

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 7

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Dev is tasked with creating a program that efficiently finds the middle element of a linked list. The program should take user input to populate the linked list by inserting each element into the front of the list and then determining the middle element.

Assist Dev, as he needs to ensure that the middle element is accurately identified from the constructed singly linked list:

If it's an odd-length linked list, return the middle element.If it's an even- length linked list, return the second middle element of the two elements.

Input Format

The first line of input consists of an integer n, representing the number of elements in the linked list.

The second line consists of n space-separated integers, representing the elements of the list.

Output Format

The first line of output displays the linked list after inserting elements at the front.

The second line displays "Middle Element: " followed by the middle element of the linked list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50

Output: 50 40 30 20 10
Middle Element: 30

Answer

```
#include
<stdio.h>
#include
<stdlib.h>     struct
Node {
   int data;
   struct Node* next;
};


struct Node* push(Node* head, int value)
{
   Node* newnode = (struct Node*) malloc(sizeof(struct
   Node)); newnode->next = head;
   newnode->data =
   value; return
   newnode;
}
int printMiddle(struct Node* head)
{
   int len = 0;
   Node* temp = head;
```

```
    while(temp != NULL)
    {
        len++;
        temp = temp->next;
    }
    int pos = len/2;
    for(int i=0; i<pos;
    i++)
    {
        head = head->next;
    }
    return head->data;
}

int main() {
    struct Node* head =
    NULL; int n;

    scanf("%d",
    &n); int value;

    for (int i = 0; i < n;
        i++) { scanf("%d",
        &value);
        head = push(head, value);
    }

    struct Node* current =
    head; while (current !=
    NULL) {
        printf("%d ",
        current->data); current =
        current->next;
    }
    printf("\n");


    int middle_element = printMiddle(head);
    printf("Middle Element: %d\n",
    middle_element);


    current = head;
    while (current != NULL) {
        struct Node* temp =
        current; current =
    }

    return
    0;
}
```
Status : Correct

Marks : *10/10*

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Your task is to create a program to manage a playlist of items. Each item is represented as a character, and you need to implement the following operations on the playlist.

Here are the main functionalities of the program:

Insert Item: The program should allow users to add items to the front and end of the playlist. Items are represented as characters.Display Playlist: The program should display the playlist containing the items that were added.

To implement this program, a doubly linked list data structure should be used, where each node contains an item character.

Input Format

The input consists of a sequence of space-separated characters, representing the items to be inserted into the doubly linked list.

The input is terminated by entering - (hyphen).

Output Format

The first line of output prints "Forward Playlist: " followed by the linked list after inserting the items at the end.

The second line prints "Backward Playlist: " followed by the linked list after inserting the items at the front.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: a b c -

Output: Forward Playlist: a b c
Backward Playlist: c b a

Answer

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
   char item;
   struct Node* next;
   struct Node* prev;
};

void insertAtEnd(Node** head, char ch) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->item = ch;
   newNode->next = NULL;
   newNode->prev = NULL;

   if (*head == NULL) {
      *head = newNode;
      return;
   }
```

```
    Node* temp = *head;
    while (temp->next != NULL)
        temp = temp->next;

    temp->next = newNode;
    newNode->prev = temp;
}

void displayForward(Node* head) {
    while (head != NULL) {
        printf("%c ", head->item);
        head = head->next;
    }
    printf("\n");
}

void displayBackward(Node* tail) {
    while (tail != NULL) {
        printf("%c ", tail->item);
        tail = tail->prev;
    }
    printf("\n");
}

void freePlaylist(Node* head) {
    Node* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
}

int main() {
    struct Node* playlist = NULL;
    char item;

    while (1) {
        scanf(" %c", &item);
        if (item == '-') {
            break;
        }
        insertAtEnd(&playlist, item);
    }

    struct Node* tail = playlist;
    while (tail->next != NULL) {
        tail = tail->next;
    }

    printf("Forward Playlist: ");
    displayForward(playlist);

    printf("Backward Playlist: ");
    displayBackward(tail);

    freePlaylist(playlist);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 2

Attempt : 2
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Moniksha, a chess coach organizing a tournament, needs a program to manage participant IDs efficiently. The program maintains a doubly linked list of IDs and offers two functions: Append to add IDs as students register, and Print Maximum ID to identify the highest ID for administrative tasks.

This tool streamlines tournament organization, allowing Moniksha to focus on coaching her students effectively.

Input Format

The first line consists of an integer n, representing the number of participant IDs to be added.

The second line consists of n space-separated integers representing the participant IDs.

Output Format

The output displays a single integer, representing the maximum participant ID.

If the list is empty, the output prints "Empty list!".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
163 137 155
Output: 163

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
   int data;
   struct Node* prev;
   struct Node* next;
} Node;

typedef struct DoublyLinkedList {
   Node* head;
   Node* tail;
} DoublyLinkedList;

void append(DoublyLinkedList* list, int data) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = NULL;

   if (list->tail == NULL) {  // If list is empty
      list->head = list->tail = newNode;
   } else {
      list->tail->next = newNode;
```

```
        newNode->prev = list->tail;
        list->tail = newNode;
    }
}

int findMax(DoublyLinkedList* list) {
    if (list->head == NULL) {
        printf("Empty list!\n");
        return -1;
    }

    Node* current = list->head;
    int maxID = current->data;

    while (current != NULL) {
        if (current->data > maxID) {
            maxID = current->data;
        }
        current = current->next;
    }

    return maxID;
}

int main() {
    DoublyLinkedList list = {NULL, NULL};

    int n, id;
    scanf("%d", &n);

    if (n < 1 || n > 20) {
        printf("Empty list!\n");
        return 0;
    }

    for (int i = 0; i < n; i++) {
        scanf("%d", &id);
        if (id < 1 || id > 10000000) {
            printf("Invalid ID!\n");
            return 0;
        }
        append(&list, id);
    }

    int maxID = findMax(&list);
    if (maxID != -1) {
        printf("%d\n", maxID);
    }

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Bob is tasked with developing a company's employee record management system. The system needs to maintain a list of employee records using a doubly linked list. Each employee is represented by a unique integer ID.

Help Bob to complete a program that adds employee records at the front, traverses the list, and prints the same for each addition of employees to the list.

Input Format

The first line of input consists of an integer N, representing the number of employees.

The second line consists of N space-separated integers, representing the employee IDs.

Output Format

For each employee ID, the program prints "Node Inserted" followed by the current state of the doubly linked list in the next line, with the data values of each node separated by spaces.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
101 102 103 104
Output: Node Inserted
101
Node Inserted
102 101
Node Inserted
103 102 101
Node Inserted
104 103 102 101

Answer

```
#include <iostream>
using namespace std;

struct node {
   int info;
   struct node* prev, * next;
};

struct node* start = NULL;


void traverse(){
   struct node*temp=start;
   while(temp!=NULL){
      printf("%d ",temp->info);
      temp=temp->next;
   }
   printf("\n");
}
```

```
void insertAtFront(int data){
   struct node*nnode=(struct node*)malloc(sizeof(node));
   nnode->info=data;
   nnode->prev=NULL;
   nnode->next=start;

   if(start!=NULL){
     start->prev=nnode;
   }
   start=nnode;
   printf("Node Inserted\n");
}



int main() {
   int n, data;
   cin >> n;
   for (int i = 0; i < n; ++i) {
      cin >> data;
      insertAtFront(data);
      traverse();
   }
   return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ravi is developing a student registration system for a college. To efficiently store and manage the student IDs, he decides to implement a doubly linked list where each node represents a student's ID.

In this system, each student's ID is stored sequentially, and the system needs to display all registered student IDs in the order they were entered.

Implement a program that creates a doubly linked list, inserts student IDs, and displays them in the same order.

Input Format

The first line contains an integer N the number of student IDs.

The second line contains N space-separated integers representing the student IDs.

Output Format

The output should display the single line containing N space-separated integers representing the student IDs stored in the doubly linked list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50
Output: 10 20 30 40 50

Answer

```
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
   int data;
   struct Node* prev;
   struct Node* next;
} Node;

typedef struct DoublyLinkedList {
   Node* head;
   Node* tail;
} DoublyLinkedList;

void append(DoublyLinkedList* list, int data) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = NULL;

   if (list->tail == NULL) {  // If the list is empty
      list->head = list->tail = newNode;
   } else {
      list->tail->next = newNode;
```

```
        newNode->prev = list->tail;
        list->tail = newNode;
    }
}

void display(DoublyLinkedList* list) {
    Node* current = list->head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}

int main() {
    DoublyLinkedList list = {NULL, NULL};

    int n, id;
    scanf("%d", &n);

    if (n < 1 || n > 10) {
        printf("Invalid input size!\n");
        return 0;
    }

    for (int i = 0; i < n; i++) {
        scanf("%d", &id);
        if (id < 1 || id > 1000000) {
            printf("Invalid ID!\n");
            return 0;
        }
        append(&list, id);
    }

    display(&list);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ashwin is tasked with developing a simple application to manage a list of items in a shop inventory using a doubly linked list. Each item in the inventory has a unique identification number. The application should allow users to perform the following operations:

Create a List of Items: Initialize the inventory with a given number of items. Each item will be assigned a unique number provided by the user and insert the elements at end of the list.

Delete an Item: Remove an item from the inventory at a specific position.

Display the Inventory: Show the list of items before and after deletion.

If the position provided for deletion is invalid (e.g., out of range), it should display an error message.

Input Format

The first line contains an integer n, representing the number of items to be initially entered into the inventory.

The second line contains n integers, each representing the unique identification number of an item separated by spaces.

The third line contains an integer p, representing the position of the item to be deleted from the inventory.

Output Format

The first line of output prints "Data entered in the list:" followed by the data values of each node in the doubly linked list before deletion.

If p is an invalid position, the output prints "Invalid position. Try again."

If p is a valid position, the output prints "After deletion the new list:" followed by the data values of each node in the doubly linked list after deletion.

Refer to the sample output for the formatting specifications.

Sample Test Case

```
Input: 4
1 2 3 4
5
Output: Data entered in the list:
 node 1 : 1
 node 2 : 2
 node 3 : 3
 node 4 : 4
Invalid position. Try again.
```

Answer

```
// You are using GCC
void DlListcreation(int n) {
  //type your code here
  struct node * temp=NULL;

  for(int i=0;i<n;i++)
  {
    temp=(node *)malloc(sizeof(node));
    scanf("%d",&temp->num);
    temp->preptr=NULL;
    temp->nextptr=NULL;

    if(stnode==NULL)
    {
      stnode=ennode=temp;
    }
    else
    {
      ennode->nextptr=temp;
      ennode=temp;
    }
  }
}

void DlListDeleteAnyNode(int pos) {
  //type your code here

  if(stnode==NULL)
  {
    return;
  }
  if(pos==1)
  {
    DlListDeleteFirstNode();
    return;
  }
  node * temp=stnode;
  for(int i=0;i<pos-2;i++)
  {
    temp=temp->nextptr;
  }
  if(temp->nextptr->nextptr==NULL)
  {
    temp->nextptr=NULL;
    return;
```

```
      }
      node * a=temp->nextptr;
      temp->nextptr=a->nextptr;
      a->nextptr->preptr=temp;

    }

    void DlListDeleteFirstNode() {
       //type your code here
       if(stnode==NULL)
       {
          return;
       }
       struct node* temp=stnode;
       if(stnode==ennode)
       {
          ennode=NULL;
          stnode=NULL;
          return;
       }
       stnode=stnode->nextptr;
       stnode->preptr=NULL;
       free(temp);
    }

    void DlListDeleteLastNode() {
       //type your code here

       if(stnode==NULL)
       {
          return;
       }
       if(stnode==ennode)
       {
          ennode=NULL;
          stnode=NULL;
          return;
       }
       struct node * temp=ennode->preptr;
       temp->nextptr=NULL;
       free(temp);
    }
```

```
void displayDlList(int m) {
  //type your code here
  if(m==1)
  {
     printf("Data entered in the list:\n");
  }
  else if(m==2)
  {
     printf("After deletion the new list:\n");
  }
  struct node * a=stnode;
  int i=1;
  while(a!=NULL)
  {
     printf("node %d : %d\n",i,a->num);
     a=a->nextptr;
 i++;
}
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a coding competition, you are assigned a task to create a program that simulates a stack using a linked list.

The program should feature a menu-driven interface for pushing an integer to stack, popping, and displaying stack elements, with robust error handling for stack underflow situations. This challenge tests your data structure skills.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Push the integer value onto the stack. If the choice is 1, the following input is a space-separated integer, representing the element to be pushed onto the stack.

Choice 2: Pop the integer from the stack.

Choice 3: Display the elements in the stack.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the stack:

If the choice is 1, push the given integer to the stack and display the following: "Pushed element: " followed by the value pushed.

If the choice is 2, pop the integer from the stack and display the following: "Popped element: " followed by the value popped.

If the choice is 2, and if the stack is empty without any elements, print "Stack is empty. Cannot pop."

If the choice is 3, print the elements in the stack: "Stack elements (top to bottom): " followed by the space-separated values.

If the choice is 3, and there are no elements in the stack, print "Stack is empty".

If the choice is 4, exit the program and display the following: "Exiting program".

If any other choice is entered, print "Invalid choice".

Refer to the sample input and output for the exact format.

Sample Test Case

```
Input: 1 3
1 4
3
2
3
4
Output: Pushed element: 3
Pushed element: 4
Stack elements (top to bottom): 4 3
Popped element: 4
Stack elements (top to bottom): 3
Exiting program
```

Answer

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
   int data;
   struct Node* next;
};

struct Node* top = NULL;



void push(int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->next = top;
   top = newNode;
   printf("Pushed element: %d\n", value);
}
```

```
void pop() {
   if (top == NULL) {
      printf("Stack is empty. Cannot pop.\n");
      return;
   }
   struct Node* temp = top;
   printf("Popped element: %d\n", top->data);
   top = top->next;
   free(temp);
}

void displayStack() {
   if (top == NULL) {
      printf("Stack is empty\n");
      return;
   }
   struct Node* temp = top;
   printf("Stack elements (top to bottom): ");
   while (temp != NULL) {
      printf("%d ", temp->data);
      temp = temp->next;
   }
   printf("\n");
}

int main() {
   int choice, value;
   do {
      scanf("%d", &choice);
      switch (choice) {
         case 1:
            scanf("%d", &value);
            push(value);
            break;
         case 2:
            pop();
            break;
         case 3:
            displayStack();
            break;
         case 4:
                printf("Exiting program\n");
                return 0;
            default:
                printf("Invalid choice\n");
        }
    } while (choice != 4);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Sanjeev is in charge of managing a library's book storage, and he wants to create a program that simplifies this task. His goal is to implement a program that simulates a stack using an array.

Help him in writing a program that provides the following functionality:

Add Book ID to the Stack (Push): You can add a book ID to the top of the book stack. Remove Book ID from the Stack (Pop): You can remove the top book ID from the stack and display its details. If the stack is empty, you cannot remove any more book IDs.Display Books ID in the Stack (Display): You can view the books ID currently on the stack.Exit the Library: You can choose to exit the program.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Push the book onto the stack. If the choice is 1, the following input is a space-separated integer, representing the ID of the book to be pushed onto the stack.

Choice 2: Pop the book ID from the stack.

Choice 3: Display the book ID in the stack.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the stack:

1. If the choice is 1, push the given book ID to the stack and display the corresponding message.
2. If the choice is 2, pop the book ID from the stack and display the corresponding message.
3. If the choice is 2, and if the stack is empty without any book ID, print "Stack Underflow"
4. If the choice is 3, print the book IDs in the stack.
5. If the choice is 3, and there are book IDs in the stack, print "Stack is empty"
6. If the choice is 4, exit the program and display the corresponding message.
7. If any other choice is entered, print "Invalid choice"

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 1 19
1 28
2
3
2
4
Output: Book ID 19 is pushed onto the stack
Book ID 28 is pushed onto the stack

```
Book ID 28 is popped from the stack
Book ID in the stack: 19
Book ID 19 is popped from the stack
Exiting the program
```

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

#define max 10

int stack[max], top = -1;

void push(int data) {
   if (top == max - 1) {
      printf("Stack overflow\n");
      return;
   }
   top = top + 1;
   stack[top] = data;
   printf("Book ID %d is pushed onto the stack\n", data);
}

void pop() {
   if (top == -1) {
      printf("Stack underflow\n");
   } else {
      int d = stack[top];
      top = top - 1;
      printf("Book ID %d is popped from the stack\n", d);
   }
}

void display() {
   if (top == -1) {
      printf("Stack is empty\n");
   } else {
      int temp = top;
      printf("Book ID in the stack:\n");
      while (temp != -1) {
         printf("%d\n", stack[temp]);
```

```
            temp--;
        }
    }
}

int main() {
    int n, data;
    do {
        scanf("%d", &n);
        switch (n) {
            case 1:
                scanf("%d", &data);
                push(data);
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting the program\n");
                break;
            default:
                printf("Invalid choice\n");
                break;
        }
    } while (n != 4);
    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Sharon is developing a programming challenge for a coding competition. The challenge revolves around implementing a character-based stack data structure using an array.

Sharon's project involves a stack that can perform the following operations:

Push a Character: Users can push a character onto the stack.Pop a Character: Users can pop a character from the stack, removing and displaying the top character.Display Stack: Users can view the current elements in the stack.Exit: Users can exit the stack operations application.

Write a program to help Sharon to implement a program that performs the given operations.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Push the character onto the stack. If the choice is 1, the following input is a space-separated character, representing the character to be pushed onto the stack.

Choice 2: Pop the character from the stack.

Choice 3: Display the characters in the stack.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the stack:

1. If the choice is 1, push the given character to the stack and display the pushed character having the prefix "Pushed: ".
2. If the choice is 2, undo the character from the stack and display the character that is popped having the prefix "Popped: ".
3. If the choice is 2, and if the stack is empty without any characters, print "Stack is empty. Nothing to pop."
4. If the choice is 3, print the elements in the stack having the prefix "Stack elements: ".
5. If the choice is 3, and there are no characters in the stack, print "Stack is empty."
6. If the choice is 4, exit the program.
7. If any other choice is entered, print "Invalid choice"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
4
Output: Stack is empty. Nothing to pop.

Answer

```
#include <stdio.h>
```

```
#include <stdbool.h>

#define MAX_SIZE 100

char items[MAX_SIZE];
int top = -1;

void initialize() {
   top = -1;
}
bool isFull() {
   return top == MAX_SIZE - 1;
}

bool isEmpty() {
   return top == -1;
}


void push(char value) {
   if (isFull()) return;
   items[++top] = value;
   printf("Pushed: %c\n", value);
}

void pop() {
   if (isEmpty()) {
      printf("Stack is empty. Nothing to pop.\n");
   } else {
      printf("Popped: %c\n", items[top--]);
   }
}

void display() {
   if (isEmpty()) {
      printf("Stack is empty.\n");
   } else {
      printf("Stack elements: ");
      for (int i = top; i >= 0; i--) {
         printf("%c ", items[i]);
      }
      printf("\n");
   }
}

int main() {
    initialize();
    int choice;
    char value;

    while (true) {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf(" %c", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                return 0;
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are a software developer tasked with building a module for a scientific calculator application. The primary function of this module is to convert infix mathematical expressions, which are easier for users to read and write, into postfix notation (also known as Reverse Polish Notation). Postfix notation is more straightforward for the application to evaluate because it removes the need for parentheses and operator precedence rules.

The scientific calculator needs to handle various mathematical expressions with different operators and ensure the conversion is correct. Your task is to implement this infix-to-postfix conversion algorithm using a stack-based approach.

Example

Input:

a+b

Output:

ab+

Explanation:

The postfix representation of (a+b) is ab+.

Input Format

The input is a string, representing the infix expression.

Output Format

The output displays the postfix representation of the given infix expression.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: a+(b*e)
Output: abe*+

Answer

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Stack {
   int top;
   unsigned capacity;
   char* array;
};

struct Stack* createStack(unsigned capacity) {
   struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack));

   if (!stack)
```

```
        return NULL;

    stack->top = -1;
    stack->capacity = capacity;
    stack->array = (char*)malloc(stack->capacity * sizeof(char));

    return stack;
}

int isEmpty(struct Stack* stack) {
    return stack->top == -1;
}

char peek(struct Stack* stack) {
    return stack->array[stack->top];
}

char pop(struct Stack* stack) {
    if (!isEmpty(stack))
        return stack->array[stack->top--];
    return '$';
}

void push(struct Stack* stack, char op) {
    stack->array[++stack->top] = op;
}



int isOperand(char ch) {
    return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z');
}

int Prec(char ch) {
    switch (ch) {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        case '^':
            return 3;
        default:
            return -1;
    }
}

void infixToPostfix(char* exp) {
    struct Stack* stack = createStack(strlen(exp));
    int i;
    for (i = 0; exp[i]; i++) {
        if (isOperand(exp[i])) {
            printf("%c", exp[i]);
        } else if (exp[i] == '(') {
            push(stack, exp[i]);
        } else if (exp[i] == ')') {
            while (!isEmpty(stack) && peek(stack) != '(') {
                printf("%c", pop(stack));
            }
            if (!isEmpty(stack) && peek(stack) != '(') {
                printf("Invalid Expression\n");
                return;
            } else {
                pop(stack);
            }
        } else {
            while (!isEmpty(stack) && peek(stack) != '(' && Prec(exp[i]) <=
Prec(peek(stack))) {
                printf("%c", pop(stack));
            }
            push(stack, exp[i]);
        }
    }
    while (!isEmpty(stack)) {
        printf("%c", pop(stack));
    }
    free(stack->array);
    free(stack);
}
```

```
int main() {
   char exp[100];
   scanf("%s", exp);

   infixToPostfix(exp);
   return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Milton is a diligent clerk at a school who has been assigned the task of managing class schedules. The school has various sections, and Milton needs to keep track of the class schedules for each section using a stack-based system.

He uses a program that allows him to push, pop, and display class schedules for each section. Milton's program uses a stack data structure, and each class schedule is represented as a character. Help him write a program using a linked list.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Push the character onto the stack. If the choice is 1, the following input is a space-separated character, representing the class schedule to be pushed onto the stack.

Choice 2: Pop class schedule from the stack

Choice 3: Display the class schedules in the stack.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the stack:

- If the choice is 1, push the given class schedule to the stack and display the following: "Adding Section: [class schedule]"
- If the choice is 2, pop the class schedule from the stack and display the following: "Removing Section: [class schedule]"
- If the choice is 2, and if the stack is empty without any class schedules, print "Stack is empty. Cannot pop."
- If the choice is 3, print the class schedules in the stack in the following: "Enrolled Sections: " followed by the class schedules separated by space.
- If the choice is 3, and there are no class schedules in the stack, print "Stack is empty"
- If the choice is 4, exit the program and display the following: "Exiting the program"
- If any other choice is entered, print "Invalid choice"

Refer to the sample output for the exact format.

Sample Test Case

```
Input: 1 d
1 h
3
2
```

```
3
4
Output: Adding Section: d
Adding Section: h
Enrolled Sections: h d
Removing Section: h
Enrolled Sections: d
Exiting program
```

Answer

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
  char data;
  struct Node* next;
};

struct Node* top = NULL;



void push(char value) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  if (newNode == NULL) {
    printf("Memory allocation failed\n");
    return;
  }
  newNode->data = value;
  newNode->next = top;
  top = newNode;
  printf("Adding Section: %c\n", value);
}

void pop() {
  if (top == NULL) {
    printf("Stack is empty. Cannot pop.\n");
    return;
  }
  struct Node* temp = top;
  printf("Removing Section: %c\n", temp->data);
  top = top->next;
```

```
    free(temp);
}

void displayStack() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return;
    }

    printf("Enrolled Sections: ");
    struct Node* temp = top;
    while (temp != NULL) {
        printf("%c ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}


int main() {
    int choice;
    char value;
    do {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf(" %c", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                displayStack();
                break;
            case 4:
                printf("Exiting program\n");
                break;
            default:
                printf("Invalid choice\n");
        }
    } while (choice != 4);
    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Guide Harish in developing a simple queue system for a customer service center. The customer service center can handle up to 25 customers at a time. The queue needs to support basic operations such as adding a customer to the queue, serving a customer (removing them from the queue), and displaying the current queue of customers.

Use an array for implementation.

Input Format

The first line of the input consists of an integer N, the number of customers arriving at the service center.

The second line consists of N space-separated integers, representing the customer IDs in the order they arrive.

Output Format

After serving the first customer in the queue, display the remaining customers in the queue.

If a dequeue operation is attempted on an empty queue, display "Underflow".

If the queue is empty, display "Queue is empty".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
101 102 103 104 105
Output: 102 103 104 105

Answer

```
// You are using GCC
#include <stdio.h>

#define MAX_SIZE 25

void dequeue(int queue[], int *size) {
    if (*size == 0) {
        printf("Underflow\n");
        printf("Queue is empty\n");
        return;
    }

    // Shift elements left after dequeue
    for (int i = 1; i < *size; i++) {
        queue[i - 1] = queue[i];
    }

    (*size)--;

    // Print remaining queue
    for (int i = 0; i < *size; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");
}

int main() {
    int N;
    scanf("%d", &N);

    if (N < 0 || N > MAX_SIZE) {
        return 0;
    }

    int queue[MAX_SIZE];

    for (int i = 0; i < N; i++) {
        scanf("%d", &queue[i]);
    }

    dequeue(queue, &N);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

You are tasked with developing a simple ticket management system for a customer support department. In this system, customers submit support tickets, which are processed in a First-In-First-Out (FIFO) order. The system needs to handle the following operations:

Ticket Submission (Enqueue Operation): New tickets are submitted by customers. Each ticket is assigned a unique identifier (represented by an integer). When a new ticket arrives, it should be added to the end of the queue.

Ticket Processing (Dequeue Operation): The support team processes tickets in the order they are received. The ticket at the front of the queue is processed first. After processing, the ticket is removed from the queue.

Display Ticket Queue: The system should be able to display the current state of the ticket queue, showing the sequence of ticket identifiers from front to rear.

Input Format

The first input line contains an integer n, the number of tickets submitted by customers.

The second line consists of a single integer, representing the unique identifier of each submitted ticket, separated by a space.

Output Format

The first line displays the "Queue: " followed by the ticket identifiers in the queue after all tickets have been submitted.

The second line displays the "Queue After Dequeue: " followed by the ticket identifiers in the queue after processing (removing) the ticket at the front.

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 6
14 52 63 95 68 49
Output: Queue: 14 52 63 95 68 49
Queue After Dequeue: 52 63 95 68 49

Answer

```
#include <stdio.h>

#define MAX_SIZE 20

void displayQueue(int queue[], int size, const char *message) {
   printf("%s", message);
   for (int i = 0; i < size; i++) {
      printf("%d ", queue[i]);
   }
   printf("\n");
}

void dequeue(int queue[], int *size) {
    if (*size == 0) {
        return;
    }

    // Shift elements left after dequeue
    for (int i = 1; i < *size; i++) {
        queue[i - 1] = queue[i];
    }
    (*size)--;
}

int main() {
    int N;
    scanf("%d", &N);

    if (N < 2 || N > MAX_SIZE) {
        return 0;
    }

    int queue[MAX_SIZE];

    for (int i = 0; i < N; i++) {
        scanf("%d", &queue[i]);
    }

    displayQueue(queue, N, "Queue: ");

    dequeue(queue, &N);

    displayQueue(queue, N, "Queue After Dequeue: ");

    return 0;
}
// You are using GCC
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

You've been assigned the challenge of developing a queue data structure using a linked list.

The program should allow users to interact with the queue by enqueuing positive integers and subsequently dequeuing and displaying elements.

Input Format

The input consists of a series of integers, one per line. Enter positive integers into the queue.

Enter -1 to terminate input.

Output Format

The output prints the space-separated dequeued elements.

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 1
2
3
4
-1
Output: Dequeued elements: 1 2 3 4

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

// Node structure
typedef struct Node {
   int data;
   struct Node* next;
} Node;

// Queue structure
```

```
typedef struct Queue {
    Node* front;
    Node* rear;
} Queue;

// Function to initialize a queue
void initQueue(Queue* q) {
    q->front = q->rear = NULL;
}

// Function to enqueue an element
void enqueue(Queue* q, int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        return;
    }
    newNode->data = value;
    newNode->next = NULL;

    if (q->rear == NULL) {
        q->front = q->rear = newNode;
    } else {
        q->rear->next = newNode;
        q->rear = newNode;
    }
}

// Function to dequeue and display elements
void dequeueAndDisplay(Queue* q) {
    printf("Dequeued elements: ");
    while (q->front != NULL) {
        Node* temp = q->front;
        printf("%d ", temp->data);
        q->front = q->front->next;
        free(temp);
    }
    printf("\n");
    q->rear = NULL;
}

int main() {
    Queue q;
    initQueue(&q);

    int value;
    while (1) {
        scanf("%d", &value);
        if (value == -1) {
            break;
        }
        enqueue(&q, value);
    }

    dequeueAndDisplay(&q);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Sharon is developing a queue using an array. She wants to provide the functionality to find the Kth largest element. The queue should support the addition and retrieval of the Kth largest element effectively. The maximum capacity of the queue is 10.

Assist her in the program.

Input Format

The first line of input consists of an integer N, representing the number of elements in the queue.

The second line consists of N space-separated integers.

The third line consists of an integer K.

Output Format

For each enqueued element, print a message: "Enqueued: " followed by the element.

The last line prints "The [K]th largest element: " followed by the Kth largest element.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
23 45 93 87 25
4

```
Output: Enqueued: 23
Enqueued: 45
Enqueued: 93
Enqueued: 87
Enqueued: 25
The 4th largest element: 25
```

Answer

```
// You are using GCC
#include <stdio.h>

#define MAX_SIZE 10

// Function to find the Kth largest element
int findKthLargest(int queue[], int size, int K) {
   // Sort the array in descending order
   for (int i = 0; i < size - 1; i++) {
      for (int j = i + 1; j < size; j++) {
         if (queue[i] < queue[j]) {
            int temp = queue[i];
            queue[i] = queue[j];
            queue[j] = temp;
         }
      }
   }
   return queue[K - 1]; // Return Kth largest element
}

int main() {
   int N;
   scanf("%d", &N);
```

```
    if (N < 1 || N > MAX_SIZE) {
        return 0;
    }

    int queue[MAX_SIZE];

    // Enqueue elements
    for (int i = 0; i < N; i++) {
        scanf("%d", &queue[i]);
        printf("Enqueued: %d\n", queue[i]);
    }

    int K;
    scanf("%d", &K);

    if (K < 4 || K > N) {
        return 0;
    }

    // Find and print the Kth largest element
    int KthLargest = findKthLargest(queue, N, K);
    printf("The %dth largest element: %d\n", K, KthLargest);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

5. Problem Statement

Amar is working on a project where he needs to implement a special type of queue that allows selective dequeuing based on a given multiple. He wants to efficiently manage a queue of integers such that only elements not divisible by a given multiple are retained in the queue after a selective dequeue operation.

Implement a program to assist Amar in managing his selective queue.

Example

Input:

5

10 2 30 4 50

5

Output:

Original Queue: 10 2 30 4 50

Queue after selective dequeue: 2 4

Explanation:

After selective dequeue with a multiple of 5, the elements that are multiples of 5 should be removed. Therefore, only 10, 30, and 50 should be removed from the queue. The updated Queue is 2 4.

Input Format

The first line contains an integer n, representing the number of elements initially present in the queue.

The second line contains n space-separated integers, representing the elements of the queue.

The third line contains an integer multiple, representing the divisor for selective dequeue operation.

Output Format

The first line of output prints "Original Queue: " followed by the space-separated elements in the queue before the dequeue operation.

The second line prints "Queue after selective dequeue: " followed by the remaining space-separated elements in the queue, after deleting elements that are the multiples of the specified number.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
10 2 30 4 50
5
Output: Original Queue: 10 2 30 4 50
Queue after selective dequeue: 2 4

Answer

```
// You are using GCC
#include <stdio.h>

#define MAX_SIZE 50
void displayQueue(int queue[], int size, const char *message) {
    printf("%s", message);
    for (int i = 0; i < size; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");
}
int selectiveDequeue(int queue[], int size, int multiple, int newQueue[]) {
    int newSize = 0;
    for (int i = 0; i < size; i++) {
        if (queue[i] % multiple != 0) {
            newQueue[newSize++] = queue[i];
        }
    }
    return newSize;
}

int main() {
    int n;
    scanf("%d", &n);

    if (n < 1 || n > MAX_SIZE) {
        return 0;
    }

    int queue[MAX_SIZE];

    for (int i = 0; i < n; i++) {
        scanf("%d", &queue[i]);
    }
    int multiple;
    scanf("%d", &multiple);

    displayQueue(queue, n, "Original Queue: ");

    int newQueue[MAX_SIZE];
    int newSize = selectiveDequeue(queue, n, multiple, newQueue);
    displayQueue(newQueue, newSize, "Queue after selective dequeue: ");

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_PAH_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Viha, a software developer, is working on a project to automate searching for a target value in a Binary Search Tree (BST). She needs to create a program that takes an integer target value as input and determines if that value is present in the BST or not.

Write a program to assist Viha.

Input Format

The first line of input consists of integers separated by spaces, which represent the elements to be inserted into the BST. The input is terminated by entering -1.

The second line consists of an integer target, which represents the target value to be searched in the BST.

Output Format

If the target value is found in the BST, print "[target] is found in the BST".

Else, print "[target] is not found in the BST"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5 3 7 1 4 6 8 -1
4
Output: 4 is found in the BST

Answer

```
#include <stdio.h>
#include <stdlib.h>


typedef struct Node {
   int data;
   struct Node* left;
   struct Node* right;
} Node;


Node* createNode(int data) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->data = data;
   newNode->left = newNode->right = NULL;
   return newNode;
}


Node* insert(Node* root, int data) {
   if (root == NULL) {
      return createNode(data);
   }
   if (data < root->data) {
      root->left = insert(root->left, data);
```

```
    } else {
        root->right = insert(root->right, data);
    }
    return root;
}


int search(Node* root, int key) {
    if (root == NULL) {
        return 0;
    }
    if (root->data == key) {
        return 1;
    }
    if (key < root->data) {
        return search(root->left, key);
    } else {
        return search(root->right, key);
    }
}


int main() {
    Node* root = NULL;
    int value;


    while (1) {
        scanf("%d", &value);
        if (value == -1) {
            break;
        }
        root = insert(root, value);
    }


    int target;
    scanf("%d", &target);


    if (search(root, target)) {
        printf("%d is found in the BST\n", target);
    } else {
        printf("%d is not found in the BST\n", target);
    }

    return 0;
}
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Joseph, a computer science student, is interested in understanding binary search trees (BST) and their node arrangements. He wants to create a program to explore BSTs by inserting elements into a tree and displaying the nodes using post-order traversal of the tree.

Write a program to help Joseph implement the program.

Input Format

The first line of input consists of an integer N, representing the number of elements to insert into the BST.

The second line consists of N space-separated integers data, which is the data to be inserted into the BST.

Output Format

The output prints N space-separated integer values after the post-order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

```
Input: 4
10 15 5 3
Output: 3 5 15 10
```

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>


typedef struct Node {
   int data;
   struct Node* left;
   struct Node* right;
} Node;


Node* createNode(int data) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->data = data;
   newNode->left = newNode->right = NULL;
   return newNode;
}


Node* insert(Node* root, int data) {
   if (root == NULL) {
      return createNode(data);
   }
   if (data < root->data) {
      root->left = insert(root->left, data);
   } else {
      root->right = insert(root->right, data);
   }
   return root;
}


void postOrderTraversal(Node* root) {
   if (root == NULL) {
      return;
   }

   postOrderTraversal(root->left);
   postOrderTraversal(root->right);
   printf("%d ", root->data);
}
```

```
int main() {
   int N;
   scanf("%d", &N);

   Node* root = NULL;
   for (int i = 0; i < N; i++) {
      int value;
      scanf("%d", &value);
      root = insert(root, value);
   }


   postOrderTraversal(root);

   return 0;
}
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Yogi is working on a program to manage a binary search tree (BST) containing integer values. He wants to implement a function that removes nodes from the tree that fall outside a specified range defined by a minimum and maximum value.

Help Yogi by writing a function that achieves this.

Input Format

The first line of input consists of an integer N, representing the number of elements to be inserted into the BST.

The second line consists of N space-separated integers, representing the elements to be inserted into the BST.

The third line consists of two space-separated integers min and max, representing the minimum value and the maximum value of the range.

Output Format

The output prints the remaining elements of the BST in an in-order traversal, after removing nodes that fall outside the specified range.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 20 12
5 15
Output: 5 10 12 15

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>


typedef struct Node {
   int data;
   struct Node* left;
   struct Node* right;
} Node;


Node* createNode(int data) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->data = data;
   newNode->left = newNode->right = NULL;
   return newNode;
}


Node* insert(Node* root, int data) {
   if (root == NULL) {
      return createNode(data);
   }
   if (data < root->data) {
        root->left = insert(root->left, data);
    } else {
        root->right = insert(root->right, data);
    }
    return root;
}


Node* trimBST(Node* root, int min, int max) {
    if (root == NULL) {
        return NULL;
    }


    root->left = trimBST(root->left, min, max);
    root->right = trimBST(root->right, min, max);


    if (root->data < min) {
        Node* rightChild = root->right;
        free(root);
        return rightChild;
    }
    if (root->data > max) {
        Node* leftChild = root->left;
        free(root);
        return leftChild;
    }

    return root;
}


void inOrderTraversal(Node* root) {
    if (root == NULL) {
        return;
    }

    inOrderTraversal(root->left);
    printf("%d ", root->data);
    inOrderTraversal(root->right);
}
```

```
int main() {
    int N, min, max;
    scanf("%d", &N);

    Node* root = NULL;
    for (int i = 0; i < N; i++) {
        int value;
        scanf("%d", &value);
        root = insert(root, value);
    }

    scanf("%d %d", &min, &max);


    root = trimBST(root, min, max);
    inOrderTraversal(root);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Aishu is participating in a coding challenge where she needs to reconstruct a Binary Search Tree (BST) from given preorder traversal data and then print the in-order traversal of the reconstructed BST.

Since Aishu is just learning about tree data structures, she needs your help to write a program that does this efficiently.

Input Format

The first line consists of an integer n, representing the number of nodes in the BST.

The second line of input contains n integers separated by spaces, which represent the preorder traversal of the BST.

Output Format

The output displays n space-separated integers, representing the in-order traversal of the reconstructed BST.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 6
10 5 1 7 40 50

Output: 1 5 7 10 40 50

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>


typedef struct Node {
   int data;
   struct Node* left;
   struct Node* right;
} Node;


Node* createNode(int data) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->data = data;
   newNode->left = newNode->right = NULL;
   return newNode;
}


Node* constructBST(int preorder[], int* index, int min, int max, int size) {
   if (*index >= size) {
      return NULL;
   }
```

```
    int value = preorder[*index];
    if (value < min || value > max) {
        return NULL;
    }

    Node* root = createNode(value);
    (*index)++;

    root->left = constructBST(preorder, index, min, value, size);
    root->right = constructBST(preorder, index, value, max, size);

    return root;
}


void inOrderTraversal(Node* root) {
    if (root == NULL) {
        return;
    }

    inOrderTraversal(root->left);
    printf("%d ", root->data);
    inOrderTraversal(root->right);
}


int main() {
    int N;
    scanf("%d", &N);

    int preorder[N];
    for (int i = 0; i < N; i++) {
        scanf("%d", &preorder[i]);
    }

    int index = 0;
    Node* root = constructBST(preorder, &index, INT_MIN, INT_MAX, N);


    inOrderTraversal(root);

    return 0;
```

}

Status : Correct ***Marks : 10/10***

5. Problem Statement

Arun is exploring operations on binary search trees (BST). He wants to write a program with an unsorted distinct integer array that represents the BST keys and construct a height-balanced BST from it.

After constructing, he wants to perform the following operations that can alter the structure of the tree and traverse them using a level-order traversal:

InsertionDeletion

Your task is to assist Arun in completing the program without any errors.

Input Format

The first line of input consists of an integer N, representing the number of initial keys in the BST.

The second line consists of N space-separated integers, representing the initial keys.

The third line consists of an integer X, representing the new key to be inserted into the BST.

The fourth line consists of an integer Y, representing the key to be deleted from the BST.

Output Format

The first line of output prints "Initial BST: " followed by a space-separated list of keys in the initial BST after constructing it in level order traversal.

The second line prints "BST after inserting a new node X: " followed by a space-separated list of keys in the BST after inserting X n level order traversal.

The third line prints "BST after deleting node Y: " followed by a space-separated list of keys in the BST after deleting Y n level order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
25 14 56 28 12
34
12
Output: Initial BST: 25 14 56 12 28
BST after inserting a new node 34: 25 14 56 12 28 34
BST after deleting node 12: 25 14 56 28 34

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
   int key;
   struct Node* left;
   struct Node* right;
} Node;

Node* createNode(int val) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->key = val;
   newNode->left = newNode->right = NULL;
   return newNode;
}

Node* insert(Node* root, int key) {
   if (root == NULL) return createNode(key);
   if (key < root->key) root->left = insert(root->left, key);
   else root->right = insert(root->right, key);
   return root;
}

Node* minValueNode(Node* node) {
   while (node && node->left) node = node->left;
```

```
    return node;
}

Node* deleteNode(Node* root, int key) {
    if (!root) return NULL;
    if (key < root->key)
        root->left = deleteNode(root->left, key);
    else if (key > root->key)
        root->right = deleteNode(root->right, key);
    else {
        if (!root->left) {
            Node* temp = root->right;
            free(root);
            return temp;
        }
        if (!root->right) {
            Node* temp = root->left;
            free(root);
            return temp;
        }
        Node* temp = minValueNode(root->right);
        root->key = temp->key;
        root->right = deleteNode(root->right, temp->key);
    }
    return root;
}

// Queue for level order traversal
typedef struct QueueNode {
    Node* treeNode;
    struct QueueNode* next;
} QueueNode;

typedef struct Queue {
    QueueNode *front, *rear;
} Queue;

Queue* createQueue() {
    Queue* q = (Queue*)malloc(sizeof(Queue));
    q->front = q->rear = NULL;
    return q;
}
```

```
void enqueue(Queue* q, Node* node) {
    QueueNode* temp = (QueueNode*)malloc(sizeof(QueueNode));
    temp->treeNode = node;
    temp->next = NULL;
    if (!q->rear) {
        q->front = q->rear = temp;
    } else {
        q->rear->next = temp;
        q->rear = temp;
    }
}

Node* dequeue(Queue* q) {
    if (!q->front) return NULL;
    QueueNode* temp = q->front;
    Node* node = temp->treeNode;
    q->front = q->front->next;
    if (!q->front) q->rear = NULL;
    free(temp);
    return node;
}

int isEmpty(Queue* q) {
    return q->front == NULL;
}

void levelOrder(Node* root) {
    if (!root) return;
    Queue* q = createQueue();
    enqueue(q, root);
    while (!isEmpty(q)) {
        Node* curr = dequeue(q);
        printf("%d ", curr->key);
        if (curr->left) enqueue(q, curr->left);
        if (curr->right) enqueue(q, curr->right);
    }
    printf("\n");
    free(q);
}

int main() {
    int N, X, Y, i, val;
    scanf("%d", &N);
    Node* root = NULL;
    for (i = 0; i < N; ++i) {
        scanf("%d", &val);
        root = insert(root, val);
    }
    scanf("%d %d", &X, &Y);

    printf("Initial BST: ");
    levelOrder(root);

    root = insert(root, X);
    printf("BST after inserting a new node %d: ", X);
    levelOrder(root);

    root = deleteNode(root, Y);
    printf("BST after deleting node %d: ", Y);
    levelOrder(root);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_PAH_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Vishnu, a math enthusiast, is given a task to explore the magic of numbers. He has an array of positive integers, and his goal is to find the integer with the highest digit sum in the sorted array using the merge sort algorithm.

You have to assist Vishnu in implementing the merge sort algorithm.

Input Format

The first line of input consists of an integer N, representing the number of elements in the array.

The second line consists of N space-separated integers, representing the array elements.

Output Format

The first line of output prints "The sorted array is: " followed by the sorted array, separated by a space.

The second line prints "The integer with the highest digit sum is: " followed by an integer representing the highest-digit sum.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
123 456 789 321 654

Output: The sorted array is: 123 321 456 654 789
The integer with the highest digit sum is: 789

Answer

```
// You are using GCC
#include <stdio.h>


int digitSum(int num) {
   int sum = 0;
   while (num > 0) {
      sum += num % 10;
      num /= 10;
   }
   return sum;
}


void merge(int arr[], int left, int mid, int right) {
   int i, j, k;
   int n1 = mid - left + 1;
   int n2 = right - mid;


   int L[n1], R[n2];


   for (i = 0; i < n1; i++)
```

```
        L[i] = arr[left + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];


    i = 0; j = 0; k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }


    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }


    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}


void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;


        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

int main() {
    int N;


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


    for (int i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }


    mergeSort(arr, 0, N - 1);


    int maxDigitSum = 0, maxDigitSumNumber = arr[0];
    for (int i = 0; i < N; i++) {
        int sum = digitSum(arr[i]);
        if (sum > maxDigitSum) {
            maxDigitSum = sum;
            maxDigitSumNumber = arr[i];
        }
    }


    printf("The sorted array is: ");
    for (int i = 0; i < N; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");


    printf("The integer with the highest digit sum is: %d\n", maxDigitSumNumber);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Alex is working on a project that involves merging and sorting two arrays. He wants to write a program that merges two arrays, sorts the merged array in ascending order, removes duplicates, and prints the sorted array without duplicates.

Help Alex to implement the program using the merge sort algorithm.

Input Format

The first line of input consists of an integer N, representing the number of elements in the first array.

The second line consists of N integers, separated by spaces, representing the elements of the first array.

The third line consists of an integer M, representing the number of elements in the second array.

The fourth line consists of M integers, separated by spaces, representing the elements of the second array.

Output Format

The output prints space-separated integers, representing the merged and sorted array in ascending order, with duplicate elements removed.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
1 2 3 4
3
3 4 5
Output: 1 2 3 4 5

Answer

```
// You are using GCC
#include <stdio.h>


void merge(int arr[], int left, int mid, int right) {
   int i, j, k;
   int n1 = mid - left + 1;
   int n2 = right - mid;

   int L[n1], R[n2];

   for (i = 0; i < n1; i++)
      L[i] = arr[left + i];
   for (j = 0; j < n2; j++)
      R[j] = arr[mid + 1 + j];

   i = 0; j = 0; k = left;
   while (i < n1 && j < n2) {
      if (L[i] <= R[j]) {
         arr[k++] = L[i++];
      } else {
         arr[k++] = R[j++];
      }
   }

   while (i < n1) {
      arr[k++] = L[i++];
   }

   while (j < n2) {
      arr[k++] = R[j++];
   }
}



void mergeSort(int arr[], int left, int right) {
   if (left < right) {
      int mid = left + (right - left) / 2;

      mergeSort(arr, left, mid);
```

```
        mergeSort(arr, mid + 1, right);

        merge(arr, left, mid, right);
    }
}


void printUniqueSortedArray(int arr[], int size) {
    printf("%d ", arr[0]);
    for (int i = 1; i < size; i++) {
        if (arr[i] != arr[i - 1]) {
            printf("%d ", arr[i]);
        }
    }
}

int main() {
    int N, M;



    scanf("%d", &N);
    int arr1[N];
    for (int i = 0; i < N; i++) {
        scanf("%d", &arr1[i]);
    }


    scanf("%d", &M);
    int arr2[M];
    for (int i = 0; i < M; i++) {
        scanf("%d", &arr2[i]);
    }


    int mergedSize = N + M;
    int mergedArray[mergedSize];
    for (int i = 0; i < N; i++) {
        mergedArray[i] = arr1[i];
    }
    for (int i = 0; i < M; i++) {
        mergedArray[N + i] = arr2[i];
    }

    mergeSort(mergedArray, 0, mergedSize - 1);

    printUniqueSortedArray(mergedArray, mergedSize);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

You are working as a programmer at a sports academy, and the academy holds various sports competitions regularly.

As part of the academy's system, you need to sort the scores of the participants in descending order using the Quick Sort algorithm.

Write a program that takes the scores of n participants as input and uses the Quick Sort algorithm to sort the scores in descending order. Your program should display the sorted scores after the sorting process.

Input Format

The first line of input consists of an integer n, which represents the number of scores.

The second line of input consists of n integers, which represent scores separated by spaces.

Output Format

Each line of output represents an iteration of the Quick Sort algorithm, displaying the elements of the array at that iteration.

After the iterations are complete, the last line of output prints the sorted scores in descending order separated by space.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

```
Input: 5
78 54 96 32 53
Output: Iteration 1: 78 54 96 53 32
Iteration 2: 96 54 78
Iteration 3: 78 54
Sorted Order: 96 78 54 53 32
```

Answer

```
// You are using GCC
#include <stdio.h>


void swap(int *a, int *b) {
   int temp = *a;
   *a = *b;
   *b = temp;
}


int partition(int arr[], int low, int high, int iteration) {
   int pivot = arr[high];
   int i = low - 1;

   for (int j = low; j < high; j++) {
      if (arr[j] >= pivot) {
         i++;
         swap(&arr[i], &arr[j]);
      }
   }
   swap(&arr[i + 1], &arr[high]);


   printf("Iteration %d: ", iteration);
   for (int k = low; k <= high; k++) {
      printf("%d ", arr[k]);
   }
   printf("\n");
```

```
    return i + 1;
}

void quickSort(int arr[], int low, int high, int *iteration) {
    if (low < high) {
        int pi = partition(arr, low, high, *iteration);
        (*iteration)++;

        quickSort(arr, low, pi - 1, iteration);
        quickSort(arr, pi + 1, high, iteration);
    }
}

int main() {
    int n;


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


    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int iteration = 1;
    quickSort(arr, 0, n - 1, &iteration);


    printf("Sorted Order: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

You're a coach managing a list of finishing times for athletes in a race. The times are stored in an array, and you need to sort this array in ascending order to determine the rankings.

You'll use the insertion sort algorithm to accomplish this.

Input Format

The first line of input contains an integer n, representing the number of athletes.

The second line contains n space-separated integers, each representing the finishing time of an athlete in seconds.

Output Format

The output prints the sorted finishing times of the athletes in ascending order.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
75 89 65 90 70
Output: 65 70 75 89 90

Answer

```
// You are using GCC
#include <stdio.h>


void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;


        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

int main() {
    int n;


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


    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }


    insertionSort(arr, n);


    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Status : Correct ***Marks : 10/10***

5. Problem Statement

You are working on an optimization task for a sorting algorithm that uses insertion sort. Your goal is to determine the efficiency of the algorithm by counting the number of swaps needed to sort an array of integers.

Write a program that takes an array as input and calculates the number of swaps performed during the insertion sort process.

Example 1:

Input:

5

2 1 3 1 2

Output:

4

Explanation:

Step 1: [2, 1, 3, 1, 2] (No swaps)

Step 2: [1, 2, 3, 1, 2] (1 swap, element 1 shifts 1 place to the left)

Step 3: [1, 2, 3, 1, 2] (No swaps)

Step 4: [1, 1, 2, 3, 2] (2 swaps; element 1 shifts 2 places to the left)

Step 5: [1, 1, 2, 2, 3] (1 swap, element 2 shifts 1 place to the left)

Total number of swaps: 1 + 2 + 1 = 4

Example 2:

Input:

7

12 15 1 5 6 14 11

Output:

10

Explanation:

Step 1: [12, 15, 1, 5, 6, 14, 11] (No swaps)

Step 2: [12, 15, 1, 5, 6, 14, 11] (1 swap, element 15 shifts 1 place to the left)

Step 3: [12, 15, 1, 5, 6, 14, 11] (No swaps)

Step 4: [1, 12, 15, 5, 6, 14, 11] (2 swaps, element 1 shifts 2 places to the left)

Step 5: [1, 5, 12, 15, 6, 14, 11] (1 swap, element 5 shifts 1 place to the left)

Step 6: [1, 5, 6, 12, 15, 14, 11] (2 swaps, element 6 shifts 2 places to the left)

Step 7: [1, 5, 6, 12, 14, 15, 11] (1 swap, element 14 shifts 1 place to the left)

Step 8: [1, 5, 6, 11, 12, 14, 15] (3 swaps, element 11 shifts 3 places to the left)

Total number of swaps: 1 + 2 + 1 + 2 + 1 + 3 = 10

Input Format

The first line of input consists of an integer n, representing the number of elements in the array.

The second line of input consists of n space-separated integers, representing the elements of the array.

Output Format

The output prints the number of swaps performed during the insertion sort process.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
2 1 3 1 2
Output: 4

Answer

```
// You are using GCC
#include <stdio.h>


int insertionSort(int arr[], int n) {
   int swapCount = 0;

   for (int i = 1; i < n; i++) {
      int key = arr[i];
      int j = i - 1;
```

```
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
            swapCount++;
        }
        arr[j + 1] = key;
    }

    return swapCount;
}

int main() {
    int n;



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


    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }


    int swapCount = insertionSort(arr, n);


    printf("%d\n", swapCount);

    return 0;
}
```

Status : Correct ***Marks : 10/10***