Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 1

Attempt : 3
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

```
#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);
```

```
        return;
    }
    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: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 2

Attempt : 2
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 v){
   struct node*newn=(struct node*)malloc(sizeof(struct node));
   newn->data=v;
   newn->next=NULL;
   if(head==NULL){
      head=newn;
      tail=newn;

   }
   else{
        tail->next=newn;
        tail=newn;
    }
}
void display_List(){
    struct node*temp=head;
    while(temp!=NULL){
        printf("%d",temp->data);
        temp=temp->next;
    }printf("\n");
}
void deleteNode(int pos){
    if(head==NULL||pos<1){
        printf("Invalid position.Deletion not possible.");
    return;

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


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: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 3

Attempt : 2
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

```
// You are using GCC
#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->value);
        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(){
    int n;
    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: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

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->next=*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;

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

```
    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: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

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)
   {
      printf("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, pos);
   traverse(head);
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

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: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

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 = current->next;
        free(temp);
    }

    return 0;
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

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 : 0

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

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 2

Attempt : 1
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>
struct Node {
   int id;
   struct Node* prev;
   struct Node* next;
};
struct DoublyLinkedList {
   struct Node* head;
   struct Node* tail;
};
void append(struct DoublyLinkedList* list, int id) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->id = id;
   newNode->prev = list->tail;
   newNode->next = NULL;
   if (list->tail != NULL) {
      list->tail->next = newNode;
   }
   list->tail = newNode;
   if (list->head == NULL) {
      list->head = newNode;
   }
}
```

```
int findMax(struct DoublyLinkedList* list) {
    if (list->head == NULL) {
        return -1; // Indicates an empty list
    }
    int maxId = list->head->id;
    struct Node* current = list->head->next;
    while (current != NULL) {
        if (current->id > maxId) {
            maxId = current->id;
        }
        current = current->next;
    }
    return maxId;
}
int main() {
    int n;
    scanf("%d", &n);
    if (n == 0) {
        printf("Empty list!\n");
        return 0;
    }
    struct DoublyLinkedList list = {NULL, NULL};
    for (int i = 0; i < n; i++) {
        int id;
        scanf("%d", &id);
        append(&list, id);
    }
    int maxId = findMax(&list);
    if (maxId == -1) {
        printf("Empty list!\n");
    } else {
        printf("%d\n", maxId);
    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

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

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

struct Node {
   int id;
   struct Node* next;
   struct Node* prev;
};
struct Node* createNode(int id) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->id = id;
   newNode->next = NULL;
   newNode->prev = NULL;
   return newNode;
}
void insertAtEnd(struct Node** head, int id) {
   struct Node* newNode = createNode(id);
   if (*head == NULL) {
      *head = newNode;
   } else {
      struct Node* temp = *head;
      while (temp->next != NULL) {
         temp = temp->next;
```

```
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
}
void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->id);
        temp = temp->next;
    }
    printf("\n");
}
int main() {
    int N;
    scanf("%d", &N);
    struct Node* head = NULL;

    for (int i = 0; i < N; i++) {
        int id;
        scanf("%d", &id);
        insertAtEnd(&head, id);
    }

    printList(head);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

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));
   if (newNode == NULL) {
      printf("Memory allocation failed\n");
      return;
   }
   newNode->data = value;
   newNode->next = top;
   top = newNode;
```

```
    printf("Pushed element: %d\n", value);
}

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

// Display operation
void displayStack() {
    if (top == NULL) {
        printf("Stack is empty\n");
    } else {
        printf("Stack elements (top to bottom): ");
        struct Node* current = top;
        while (current != NULL) {
            printf("%d ", current->data);
            current = current->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: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

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 100

int stack[MAX];
int top = -1;

void push(int bookID) {
    if (top >= MAX - 1) {
        printf("Stack Overflow: Cannot push Book ID %d\n", bookID);
        return;
    }
    stack[++top] = bookID;
    printf("Book ID %d is pushed onto the stack\n", bookID);
}

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

void display() {
    if (top == -1) {
        printf("stack is empty \n");
    } else {
        printf("Book ID in the stack: ");
        for (int i = top; i >= 0; i--) {
            printf("%d\n", stack[i]);
        }
    }
}
```

```
int main() {
    int choice, bookID;

    while (1) {
        if (scanf("%d", &choice) != 1)
            break;

        switch (choice) {
            case 1:
                if (scanf("%d", &bookID) != 1) break;
                push(bookID);
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting the program\n");
                exit(0);
            default:
                printf("Invalid choice\n");
                printf("Exiting the program\n");
                exit(0);
        }
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

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;
}

// You are using GCC
void push(char ch) {
   if (top >= MAX_SIZE - 1) {
      printf("Stack overflow. Cannot push more elements.\n");
      return;
   }
   items[++top] = ch;
   printf("Pushed: %c\n", ch);
}

void pop() {
   if (top == -1) {
      printf("Stack is empty. Nothing to pop.\n");
   } else {
      printf("Popped: %c\n", items[top--]);
   }
}
void display() {
   if (top == -1) {
      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: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 4

Attempt : 2
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) {
    if (ch == '+' || ch == '-') return 1;
    if (ch == '*' || ch == '/') return 2;
    if (ch == '^') return 3;
    return 0;
}
void infixToPostfix(char* exp) {
    struct Stack* stack = createStack(strlen(exp));
    if (!stack) {
        printf("Memory allocation failed\n");
        return;
    }
    char output[100];
    int k = 0;
    for (int i = 0; exp[i]; i++) {
        char current = exp[i];
    if (isOperand(current)) {
        output[k++] = current;
    }

    else if (current == '(') {
        push(stack, current);
    }

    else if (current == ')') {
        while (!isEmpty(stack) && peek(stack) != '(') {
            output[k++] = pop(stack);
        }
        pop(stack);
    }

    else {
        while (!isEmpty(stack) && Prec(current) <= Prec(peek(stack))) {
            output[k++] = pop(stack);
        }
        push(stack, current);
    }
  }

  while (!isEmpty(stack)) {
    output[k++] = pop(stack);
  }
  output[k] = '\0';
  printf("%s\n", output);
  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: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

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

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

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

struct Node* top = NULL;

void push(char value) {
   struct Node* nnode = (struct Node*)malloc(sizeof(struct Node));
   nnode->data = value;
   nnode->next = top;
   top = nnode;
   printf("Adding Section: %c\n", value);
}

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

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

```
    } else {
        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: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Imagine a bustling coffee shop, where customers are placing their orders for their favorite coffee drinks. The cafe owner Sheeren wants to efficiently manage the queue of coffee orders using a digital system. She needs a program to handle this queue of orders.

You are tasked with creating a program that implements a queue for coffee orders. Each character in the queue represents a customer's coffee order, with 'L' indicating a latte, 'E' indicating an espresso, 'M' indicating a macchiato, 'O' indicating an iced coffee, and 'N' indicating a nabob.

Customers can place orders and enjoy their delicious coffee drinks.

Input Format

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

Choice 1: Enqueue the coffee order into the queue. If the choice is 1, the following input is a space-separated character ('L', 'E', 'M', 'O', 'N').

Choice 2: Dequeue a coffee order from the queue.

Choice 3: Display the orders in the queue.

Choice 4: Exit the program.

Output Format

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

If the choice is 1:

1. Insert the given order into the queue and display "Order for [order] is enqueued." where [order] is the coffee order that is inserted.
2. If the queue is full, print "Queue is full. Cannot enqueue more orders."

If the choice is 2:

1. Dequeue a character from the queue and display "Dequeued Order: " followed by the corresponding order that is dequeued.
2. If the queue is empty without any orders, print "No orders in the queue."

If the choice is 3:

1. The output prints "Orders in the queue are: " followed by the space-separated orders present in the queue.
2. If there are no orders in the queue, print "Queue is empty. No orders available."

If the choice is 4:

1. Exit the program and print "Exiting program"

If any other choice is entered, the output prints "Invalid option."

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

Sample Test Case

Input: 1 L
1 E
1 M
1 O
1 N
1 O
3
2
3
4
Output: Order for L is enqueued.
Order for E is enqueued.
Order for M is enqueued.
Order for O is enqueued.
Order for N is enqueued.
Queue is full. Cannot enqueue more orders.
Orders in the queue are: L E M O N
Dequeued Order: L
Orders in the queue are: E M O N
Exiting program

Answer

```
#include <stdio.h>
#define MAX_SIZE 5

char orders[MAX_SIZE];
int front = -1;
int rear = -1;

void initializeQueue() {
   front = -1;
   rear = -1;
}

int isEmpty() {
 return front == -1;
}
int isFull() {
 return ((rear + 1) % MAX_SIZE) == front;
}
int enqueue(char order) {
 if (isFull()) {
 printf("Queue is full. Cannot enqueue more orders.\n");
 return 0;
 }
 if (isEmpty()) {
 front = rear = 0;
 } else {
 rear = (rear + 1) % MAX_SIZE;
 }
 orders[rear] = order;
 printf("Order for %c is enqueued.\n", order);
 return 1;
}
int dequeue() {
 if (isEmpty()) {
 printf("No orders in the queue.\n");
 return 0;
 }
 char c = orders[front];
 if (front == rear) {
 front = rear = -1;
 } else {
 front = (front + 1) % MAX_SIZE;
 }
 printf("Dequeued Order: %c\n", c);
 return 1;
}
void display() {
 if (isEmpty()) {
 printf("Queue is empty. No orders available.\n");
 return;
 }
 printf("Orders in the queue are: ");
 int i = front;
 while (1) {
 printf("%c", orders[i]);
 if (i == rear) break;
 printf(" ");
 i = (i + 1) % MAX_SIZE;
 }
 printf("\n");
}

int main() {
   char order;
   int option;
   initializeQueue();
   while (1) {
      if (scanf("%d", &option) != 1) {
         break;
      }
      switch (option) {
         case 1:
            if (scanf(" %c", &order) != 1) {
               break;
            }
            if (enqueue(order)) {
            }
            break;
         case 2:
            dequeue();
            break;
         case 3:
            display();
            break;
         case 4:
            printf("Exiting program");
            return 0;
         default:
            printf("Invalid option.\n");
            break;
      }
   }
   return 0;
}
```

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

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a bustling IT department, staff regularly submit helpdesk tickets to request technical assistance. Managing these tickets efficiently is vital for providing quality support.

Your task is to develop a program that uses an array-based queue to handle and prioritize helpdesk tickets based on their unique IDs.

Implement a program that provides the following functionalities:

Enqueue Helpdesk Ticket: Add a new helpdesk ticket to the end of the queue. Provide a positive integer representing the ticket ID for the new ticket.Dequeue Helpdesk Ticket: Remove and process the next helpdesk ticket from the front of the queue. The program will display the ticket ID of the processed ticket.Display Queue: Display the ticket IDs of all the helpdesk tickets currently in the queue.

Input Format

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

Choice 1: Enqueue the ticket ID into the queue. If the choice is 1, the following input is a space-separated integer, representing the ticket ID to be enqueued into the queue.

Choice 2: Dequeue a ticket from the queue.

Choice 3: Display the ticket IDs in the queue.

Choice 4: Exit the program.

Output Format

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

If the choice is 1:

1. Insert the given ticket ID into the queue and display "Helpdesk Ticket ID [id] is enqueued." where [id] is the ticket ID that is inserted.
2. If the queue is full, print "Queue is full. Cannot enqueue."

If the choice is 2:

1. Dequeue a ticket ID from the queue and display "Dequeued Helpdesk Ticket ID: " followed by the corresponding ID that is dequeued.
2. If the queue is empty without any elements, print "Queue is empty."

If the choice is 3:

1. The output prints "Helpdesk Ticket IDs in the queue are: " followed by the space-separated ticket IDs present in the queue.
2. If there are no elements in the queue, print "Queue is empty."

If the choice is 4:

1. Exit the program and print "Exiting the program"

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

Refer to the sample output for formatting specifications.

Sample Test Case

```
Input: 1 101
1 202
1 203
1 204
1 205
1 206
3
2
3
4
Output: Helpdesk Ticket ID 101 is enqueued.
Helpdesk Ticket ID 202 is enqueued.
Helpdesk Ticket ID 203 is enqueued.
Helpdesk Ticket ID 204 is enqueued.
Helpdesk Ticket ID 205 is enqueued.
Queue is full. Cannot enqueue.
Helpdesk Ticket IDs in the queue are: 101 202 203 204 205
Dequeued Helpdesk Ticket ID: 101
Helpdesk Ticket IDs in the queue are: 202 203 204 205
Exiting the program
```

Answer

```
#include <stdio.h>
#define MAX_SIZE 5

int ticketIDs[MAX_SIZE];
int front = -1;
int rear = -1;
int lastDequeued;

void initializeQueue() {
   front = -1;
   rear = -1;
}
```

```
int isEmpty() {
 return front == -1;
}
int isFull() {
 return ((rear + 1) % MAX_SIZE) == front;
}
int enqueue(int ticketID) {
 if (isFull()) {
 printf("Queue is full. Cannot enqueue.\n");
 return 0;
 }
 if (isEmpty()) {
 front = rear = 0;
 } else {
 rear = (rear + 1) % MAX_SIZE;
 }
 ticketIDs[rear] = ticketID;
 printf("Helpdesk Ticket ID %d is enqueued.\n", ticketID);
 return 1;
}
int dequeue() {
 if (isEmpty()) {
 return 0;
 }
 lastDequeued = ticketIDs[front];
 if (front == rear) {
 front = rear = -1;
 } else {
 front = (front + 1) % MAX_SIZE;
 }
 return 1;
}
void display() {
 if (isEmpty()) {
 printf("Queue is empty.\n");
 return;
 }
 printf("Helpdesk Ticket IDs in the queue are: ");
 int i = front;
 while (1) {
 printf("%d", ticketIDs[i]);
 if (i == rear) break;
 printf(" ");
 i = (i + 1) % MAX_SIZE;
 }
 printf("\n");
}

int main() {
    int ticketID;
    int option;
    initializeQueue();
    while (1) {
        if (scanf("%d", &option) == EOF) {
            break;
        }
        switch (option) {
            case 1:
                if (scanf("%d", &ticketID) == EOF) {
                    break;
                }
                enqueue(ticketID);
                break;
            case 2:
                if (dequeue()) {
                    printf("Dequeued Helpdesk Ticket ID: %d\n", lastDequeued);
                } else {
                    printf("Queue is empty.\n");
                }
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting the program\n");
                return 0;
            default:
                printf("Invalid option.\n");
                break;
        }
    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Write a program to implement a queue using an array and pointers. The program should provide the following functionalities:

Insert an element into the queue.Delete an element from the queue.Display the elements in the queue.

The queue has a maximum capacity of 5 elements. If the queue is full and an insertion is attempted, a "Queue is full" message should be displayed. If the queue is empty and a deletion is attempted, a "Queue is empty" message should be displayed.

Input Format

Each line contains an integer representing the chosen option from 1 to 3.

Option 1: Insert an element into the queue followed by an integer representing the element to be inserted, separated by a space.

Option 2: Delete an element from the queue.

Option 3: Display the elements in the queue.

Output Format

For option 1 (insertion):-

1. The program outputs: "<data> is inserted in the queue." if the data is successfully inserted.
2. "Queue is full." if the queue is already full and cannot accept more elements.

For option 2 (deletion):-

1. The program outputs: "Deleted number is: <data>" if an element is successfully deleted and returns the value of the deleted element.
2. "Queue is empty." if the queue is empty no elements can be deleted.

For option 3 (display):-

1. The program outputs: "Elements in the queue are: <element1> <element2> ... <elementN>" where <element1>, <element2>, ..., <elementN> represent the elements present in the queue.
2. "Queue is empty." if the queue is empty no elements can be displayed.

For invalid options, the program outputs: "Invalid option."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1 10

3
5
Output: 10 is inserted in the queue.
Elements in the queue are: 10
Invalid option.

Answer

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

#define max 5

int queue[max];
int front = -1, rear = -1;
int insertq(int *data) {
 if ((rear + 1) % max == front) return 0;
 if (front == -1) {
 front = rear = 0;
 } else {
 rear = (rear + 1) % max;
 }
 queue[rear] = *data;
 return 1;
}
int delq() {
 int val;
 if (front == -1) {
 printf("Queue is empty.\n");
 return -1;
 }
 val = queue[front];
 if (front == rear) {
 front = rear = -1;
 } else {
 front = (front + 1) % max;
 }
 printf("Deleted number is: %d\n", val);
 return val;
}
void display() {
 int i;
 if (front == -1) {
 printf("Queue is empty.\n");
 return;
 }
 printf("Elements in the queue are: ");
 i = front;
 do {
 printf("%d ", queue[i]);
 i = (i + 1) % max;
 } while (i != (rear + 1) % max);
 printf("\n");
}

int main()
{
   int data, reply, option;
   while (1)
   {
      if (scanf("%d", &option) != 1)
         break;
      switch (option)
      {
         case 1:
            if (scanf("%d", &data) != 1)
               break;
            reply = insertq(&data);
            if (reply == 0)
               printf("Queue is full.\n");
            else
               printf("%d is inserted in the queue.\n", data);
            break;
         case 2:
            delq();  //    Called without arguments
            break;
         case 3:
            display();
            break;
         default:
            printf("Invalid option.\n");
            break;
      }
   }
   return 0;
```

```
}
```

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

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 4

Attempt : 2
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In an office setting, a print job management system is used to efficiently handle and process print jobs. The system is implemented using a queue data structure with an array.

The program provides the following operations:

Enqueue Print Job: Add a print job with a specified number of pages to the end of the queue.Dequeue Print Job: Remove and process the next print job in the queue.Display Queue: Display the print jobs in the queue

The program should ensure that print jobs are processed in the order they are received.

Input Format

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

Choice 1: Enqueue the print job into the queue. If the choice is 1, the following input is a space-separated integer, representing the pages to be enqueued into the queue.

Choice 2: Dequeue a print job from the queue.

Choice 3: Display the print jobs in the queue.

Choice 4: Exit the program.

Output Format

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

If the choice is 1:

1. Insert the given page into the queue and display "Print job with [page] pages is enqueued." where [page] is the number of pages that are inserted.
2. If the queue is full, print "Queue is full. Cannot enqueue."

If the choice is 2:

1. Dequeue a page from the queue and display "Processing print job: [page] pages" where [page] is the corresponding page that is dequeued.
2. If the queue is empty without any elements, print "Queue is empty."

If the choice is 3:

1. The output prints "Print jobs in the queue: " followed by the space-separated pages present in the queue.
2. If there are no elements in the queue, print "Queue is empty."

If the choice is 4:

1. Exit the program and print "Exiting program"

If any other choice is entered, the output prints "Invalid option."

Refer to the sample output for the formatting specifications.

Sample Test Case

```
Input: 1
10
1
20
1
30
1
40
1
50
1
60
3
2
3
4
Output: Print job with 10 pages is enqueued.
Print job with 20 pages is enqueued.
Print job with 30 pages is enqueued.
Print job with 40 pages is enqueued.
Print job with 50 pages is enqueued.
Queue is full. Cannot enqueue.
Print jobs in the queue: 10 20 30 40 50
Processing print job: 10 pages
Print jobs in the queue: 20 30 40 50
Exiting program
```

Answer

```
void enqueue(int);
void dequeue();
void display();
void display(){
 if(front==-1 && rear==-1){
 printf("Queue is empty.\n");
 }
 else{
 printf("Print jobs in the queue: ");
 int temp=front;
 while(temp!=rear+1){
 printf("%d ",queue[temp]);
 temp++;
 }
 printf("\n");
 }
}
void dequeue(){
 if(front==-1 && rear==-1){
 printf("Queue is empty.\n");
 }
 else if(front==rear){
 printf("Processing print job: %d pages\n",queue[front]);
 front=-1;
 rear=-1;
 }
 else{
 printf("Processing print job: %d pages\n",queue[front]);
 front++;
 }
}
void enqueue(int data){
 if(front==-1 && rear == -1){
 front =0;
 rear=0;
 queue[rear]=data;
 printf("Print job with %d pages is enqueued.\n",data);
 }
 else if(rear==MAX_SIZE-1){
 printf("Queue is full. Cannot enqueue.\n");
 }
 else{
 rear++;
 queue[rear]=data;
 printf("Print job with %d pages is enqueued.\n",data);
 }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are tasked with implementing basic operations on a queue data structure using a linked list.

You need to write a program that performs the following operations on a queue:

Enqueue Operation: Implement a function that inserts an integer element at the rear end of the queue.Print Front and Rear: Implement a function that prints the front and rear elements of the queue. Dequeue Operation: Implement a function that removes the front element from the queue.

Input Format

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

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

Output Format

The first line prints "Front: X, Rear: Y" where X is the front and Y is the rear elements of the queue.

The second line prints the message indicating that the dequeue operation (front element removed) is performed: "Performing Dequeue Operation:".

The last line prints "Front: M, Rear: N" where M is the front and N is the rear elements after the dequeue operation.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
12 56 87 23 45

Output: Front: 12, Rear: 45
Performing Dequeue Operation:
Front: 56, Rear: 45

Answer

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

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

struct Node* front = NULL;
struct Node* rear = NULL;

void enqueue(int d) {
 struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
 if (!newNode) exit(1);
 newNode->data = d;
 newNode->next = NULL;
 if (front == NULL) {
 front = rear = newNode;
 } else {
 rear->next = newNode;
 rear = newNode;
 }
}
void printFrontRear() {
 if (front == NULL) return;
 printf("Front: %d, Rear: %d\n", front->data, rear->data);
}
void dequeue() {
 if (front == NULL) return;
 struct Node* tmp = front;
 front = front->next;
 free(tmp);
}

int main() {
    int n, data;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        enqueue(data);
    }
    printFrontRear();
    printf("Performing Dequeue Operation:\n");
    dequeue();
    printFrontRear();
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John is learning about Binary Search Trees (BST) in his computer science class. He wants to create a program that allows users to delete a node with a given value from a BST and print the remaining nodes using an in-order traversal.

Implement a function to help him delete a node with a given value from a BST.

Input Format

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

The second line consists of N space-separated integers, representing the values of the BST nodes.

The third line consists of an integer V, which is the value to delete from the BST.

Output Format

The output prints the space-separated values in the BST in an in-order traversal, after the deletion of the specified value.

If the specified value is not available in the tree, print the given input values in-order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 2 7
15
Output: 2 5 7 10

Answer

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

struct TreeNode {
   int data;
   struct TreeNode* left;
   struct TreeNode* right;
};

struct TreeNode* createNode(int key) {
   struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct
TreeNode));
   newNode->data = key;
   newNode->left = newNode->right = NULL;
   return newNode;
}
struct TreeNode* insert(struct TreeNode* root, int key) {
   struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct
TreeNode));
```

```
    if(root==NULL){
       newNode->data=key;
       newNode->left=NULL;
       newNode->right=NULL;
       root=newNode;
    }
    else if(key<root->data)
    root->left=insert(root->left,key);
    else if(key>root->data)
    root->right=insert(root->right,key);
    return root;
}

struct TreeNode* findMin(struct TreeNode* root) {
    if(root!=NULL){
       while(root->left!=NULL)
       root=root->left;
       return root;
    }
}

struct TreeNode* deleteNode(struct TreeNode* root, int key) {
       struct TreeNode* temp = (struct TreeNode*)malloc(sizeof(struct
TreeNode));
       if(root==NULL){
          return root;
       }
       else if(key<root->data)
       root->left=deleteNode(root->left,key);
       else if(key>root->data)
       root->right=deleteNode(root->right,key);
       else if(root->left && root->right)
       {
          temp=findMin(root->right);
          root->data=temp->data;
          root->right=deleteNode(root->right,root->data);
       }
       else{
          temp=root;
          if(root->left==NULL)
          root=root->right;
          else if(root->right==NULL)
```

```
            root=root->left;
            free(temp);
        }
        return root;
}

void inorderTraversal(struct TreeNode* root) {
    if(root!=NULL)
    {
        inorderTraversal(root->left);
        printf("%d ",root->data);
        inorderTraversal(root->right);
    }
}

int main()
{
    int N, rootValue, V;
    scanf("%d", &N);
    struct TreeNode* root = NULL;
    for (int i = 0; i < N; i++) {
        int key;
        scanf("%d", &key);
        if (i == 0) rootValue = key;
        root = insert(root, key);
    }
    scanf("%d", &V);
    root = deleteNode(root, V);
    inorderTraversal(root);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Mike is learning about Binary Search Trees (BSTs) and wants to implement various operations on them. He wants to write a basic program for creating a BST, inserting nodes, and printing the tree in the pre-order traversal.

Write a program to help him solve this program.

Input Format

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

The second line consists of N space-separated integers, representing the values to insert into the BST.

Output Format

The output prints the space-separated values of the BST in the pre-order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

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

Answer

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

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

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

struct Node* insert(struct Node* root, int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if(root==NULL){
      newNode->data=value;
      newNode->left=NULL;
      newNode->right=NULL;
      root=newNode;
   }
   else if(value<root->data)
   root->left=insert(root->left,value);
   else if(value>root->data)
   root->right=insert(root->right,value);
```

```
    return root;
}

void printPreorder(struct Node* node) {
    if(node!=NULL)
    {
        printf("%d ",node->data);
        printPreorder(node->left);
        printPreorder(node->right);

    }
}

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

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

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

    printPreorder(root);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are required to implement basic operations on a Binary Search Tree (BST), like insertion and searching.

Insertion: Given a list of integers, construct a Binary Search Tree by repeatedly inserting each integer into the tree according to the rules of a BST.

Searching: Given an integer, search for its presence in the constructed Binary Search Tree. Print whether the integer is found or not.

Write a program to calculate this efficiently.

Input Format

The first line of input consists of an integer n, representing the number of nodes in the binary search tree.

The second line consists of the values of the nodes, separated by space as integers.

The third line consists of an integer representing, the value that is to be searched.

Output Format

The output prints, "Value <value> is found in the tree." if the given value is present, otherwise it prints: "Value <value> is not found in the tree."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
8 3 10 1 6 14 23
6
Output: Value 6 is found in the tree.

Answer

```
struct Node* insertNode(struct Node* root, int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if(root==NULL){
      newNode->data=value;
      newNode->left=NULL;
      newNode->right=NULL;
      return newNode;
   }
   else if(value<root->data)
   root->left=insertNode(root->left,value);
   else if(value>root->data)
   root->right=insertNode(root->right,value);
   return root;
}
struct Node* searchNode(struct Node* root, int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if(root==NULL){
      return NULL;
   }
```

```
    else if(value<root->data)
    return searchNode(root->left,value);
    else if(value>root->data)
    return searchNode(root->right,value);
    else
    return root;
}
```

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

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 4

Attempt : 2
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John, a computer science student, is learning about binary search trees (BST) and their properties. He decides to write a program to create a BST, display it in post-order traversal, and find the minimum value present in the tree.

Help him by implementing 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 first line of output prints the space-separated elements of the BST in post-order traversal.

The second line prints the minimum value found in the BST.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
5 10 15
Output: 15 10 5
The minimum value in the BST is: 5

Answer

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

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

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

// You are using GCC
struct Node* insert(struct 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 displayTreePostOrder(struct Node* root) {
    //Type your code here
    if(root==NULL)return;
    displayTreePostOrder(root->left);
    displayTreePostOrder(root->right);
    printf("%d",root->data);

}

int findMinValue(struct Node* root) {
    //Type your code here
    Node*current=root;
    while(current && current->left!=NULL){
        current=current->left;
    }
    return current->data;
}

int main() {
    struct Node* root = NULL;
    int n, data;
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        root = insert(root, data);
    }

    displayTreePostOrder(root);
    printf("\n");

    int minValue = findMinValue(root);
    printf("The minimum value in the BST is: %d", minValue);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Baskar A
Email: 240701074@rajalakshmi.edu.in
Roll no: 240701074
Phone: 7397553517
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In his computer science class, John is learning about Binary Search Trees (BST). He wants to build a BST and find the maximum value in the tree.

Help him by writing a program to insert nodes into a BST and find the maximum value in the tree.

Input Format

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

The second line consists of N space-separated integers, representing the values of the nodes to insert into the BST.

Output Format

The output prints the maximum value in the BST.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 2 7

Output: 15

Answer

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

struct TreeNode {
   int data;
   struct TreeNode* left;
   struct TreeNode* right;
};
struct TreeNode* createNode(int key) {
   struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct
TreeNode));
   newNode->data = key;
   newNode->left = newNode->right = NULL;
   return newNode;
}
struct TreeNode* insert(struct TreeNode* root, int data) {
   if (root == NULL) {
      return createNode(data);
   }
   if (data < root->data) {
      root->left = insert(root->left, data);
   } else if (data > root->data) {
      root->right = insert(root->right, data);
   }
   return root;
}
int findMax(struct TreeNode* root) {
   while (root->right != NULL) {
        root = root->right;
    }
    return root->data;
}

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

    struct TreeNode* root = NULL;

    for (int i = 0; i < N; i++) {
        int key;
        scanf("%d", &key);
        if (i == 0) rootValue = key;
        root = insert(root, key);
    }

    int maxVal = findMax(root);
    if (maxVal != -1) {
        printf("%d", maxVal);
    }

    return 0;
}
```

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