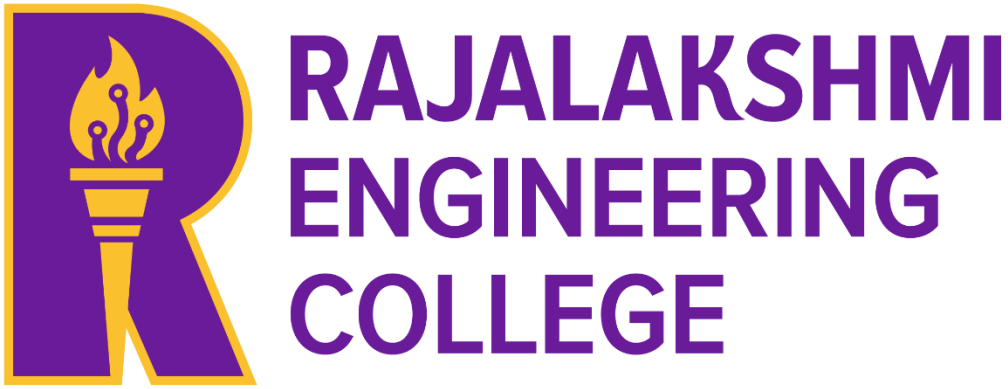
RAJALAKSHMI ENGINEERING COLLEGE

(An Autonomous Institution)

RAJALAKSHMI NAGAR, THANDALAM- 602 105



CS23231 - DATA STRUCTURES

LABORATORY RECORD NOTEBOOK

NAME: ……

YEAR/SEMESTER: ……

BRANCH/SECTION: ……

REGISTER NO: ……

COLLEGE ROLL NO: ……

ACADEMIC YEAR: 20….. -20…..

RAJALAKSHMI ENGINEERING COLLEGE

(An Autonomous Institution)

RAJALAKSHMI NAGAR, THANDALAM- 602 105

BONAFIDE CERTIFICATE

NAME: …… **BRANCH/SECTION:** ……

ACADEMIC YEAR: 20….. -20….. **SEMESTER:** ……..

REGISTER NO:

Certified that this is a Bonafide record of work done by the above student in the **CS23231 - DATA STRUCTURES** during the year 20 - 20

Signature of Faculty In-charge

Submitted for the Practical Examination Held on: ……..

Internal Examiner **External Examiner**

INDEX

EX.NO	DATE	NAME OF THE EXPERIMENT	GITHUB QR CODE	PAGE NO	STAFF SIGN
1		Implementation of Single LinkedList (Insertion, Deletion, and Display).			
2		Implementation of Doubly LinkedList (Insertion, Deletion, and Display).			
3		Implementation of a Stack using an Array and a linked implementation.			
4		Implementation of a Queue using an Array and a linked implementation.			
5		Implementation of Binary Search Tree and perform Tree Traversal Techniques.			
6		Program to perform Sorting (Quick sort, Merge sort)			
7		Program to perform Hashing (Linear Probing, Rehashing)			

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_MCQ

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : MCQ

1. The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function.

What should be added in place of "/*ADD A STATEMENT HERE*/", so that the function correctly reverses a linked list?

```
struct node {
   int data;
   struct node* next;
};
static void reverse(struct node** head_ref) {
   struct node* prev   = NULL;
   struct node* current = *head_ref;
   struct node* next;
   while (current != NULL) {
       next  = current->next;
```

```
    current->next = prev;
    prev = current;
    current = next;
  }
  /*ADD A STATEMENT HERE*/
}
```

Answer

*head_ref = prev;

Status : Correct **Marks : 1/1**

2. The following function takes a singly linked list of integers as a parameter and rearranges the elements of the lists.

The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node {
  int value;
  struct node* next;
};

void rearrange (struct node* list) {
  struct node *p,q;
  int temp;
  if (! List || ! list->next) return;
  p=list; q=list->next;
  while(q) {
    temp=p->value; p->value=q->value;
    q->value=temp;p=q->next;
    q=p?p->next:0;
  }
}
```

Answer

2, 1, 4, 3, 6, 5, 7

Status : Correct **Marks : 1/1**

3. Linked lists are not suitable for the implementation of?

Answer

Binary search

Status : Correct ***Marks : 1/1***

4. In a singly linked list, what is the role of the "tail" node?

Answer

It stores the last element of the list

Status : Correct ***Marks : 1/1***

5. Consider an implementation of an unsorted singly linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operations can be implemented in O(1) time?

i) Insertion at the front of the linked list

ii) Insertion at the end of the linked list

iii) Deletion of the front node of the linked list

iv) Deletion of the last node of the linked list

Answer

I and III

Status : Correct ***Marks : 1/1***

6. Consider the singly linked list: 15 -> 16 -> 6 -> 7 -> 17. You need to delete all nodes from the list which are prime.

What will be the final linked list after the deletion?

Answer

15 -> 16 -> 6

Status : Correct **Marks : 1/1**

7. Given a pointer to a node X in a singly linked list. If only one point is given and a pointer to the head node is not given, can we delete node X from the given linked list?

Answer

Possible if X is not last node.

Status : Correct **Marks : 1/1**

8. Which of the following statements is used to create a new node in a singly linked list?

```
struct node {
   int data;
   struct node * next;
}
typedef struct node NODE;
NODE *ptr;
```

Answer

ptr = (NODE*)malloc(sizeof(NODE));

Status : Correct **Marks : 1/1**

9. Consider the singly linked list: 13 -> 4 -> 16 -> 9 -> 22 -> 45 -> 5 -> 16 -> 6, and an integer K = 10, you need to delete all nodes from the list that are less than the given integer K.

What will be the final linked list after the deletion?

Answer

13 -> 16 -> 22 -> 45 -> 16

Status : Correct **Marks : 1/1**

10. Given the linked list: 5 -> 10 -> 15 -> 20 -> 25 -> NULL. What will be the output of traversing the list and printing each node's data?

Answer

5 10 15 20 25

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

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

```
#include<stdio.h>
int main(){
    int n,m,co,exp,sum=0;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d %d",&co,&exp);
        sum+=co;
    }
    scanf("%d",&m);
    for(int i;i<m;i++){
        scanf("%d %d",&co,&exp);
        sum+=co;
    }
    printf("%d\n",sum);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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 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=newnode;
        tail=newnode;
```

```
    }
}
void displaylist(){
    struct node*temp=head;
    while(temp!=NULL){
        printf("%d ",temp->data);
        temp=temp->next;
    }printf("\n");
}
void deleteNode(int position){
    if(head==NULL){
        printf("Invalid position. Deletion not possible.\n");
        return ;
    }
    struct node*temp=head;
    if(position==1){
        head=head->next;
        free(temp);
        displaylist();
        return;
    }
    struct node*prev=NULL;
    for(int i=1;temp!=NULL && i<position;i++){
        prev=temp;
        temp=temp->next;
    }
    if(temp==NULL){
        printf("Invalid position. Deletion not possible.\n");
        return;
    }
    prev->next=temp->next;
    if(temp==tail){
        tail=prev;
    }free(temp);
    displaylist();
}

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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
   char data;
   struct node*next;
}node;
node*createNode(char data){
   node*newnode=(node*)malloc(sizeof(node));
   newnode->data=data;
   newnode->next=NULL;
   return newnode;
```

```
}
void insertAfter(node*head,int index,char newchar){
    node*temp=head;
    int count=0;
    while(temp!=NULL && count<index){
        temp=temp->next;
        count++;
    }if(temp==NULL){
        printf("Invalid index\n");
        return ;
    }node*newnode=createNode(newchar);
    newnode->next=temp->next;
    temp->next=newnode;
}
void printlist(node*head){
    node*temp=head;
    printf("Updated list: ");
    while(temp!=NULL){
        printf("%c ",temp->data);
        temp=temp->next;
    }printf("\n");
}
int main(){
    int n,index;
    char newchar;
    scanf("%d",&n);
    node*head=NULL,*tail=NULL;
    for(int i=0;i<n;i++){
        char ch;
        scanf(" %c",&ch);
        node*newnode=createNode(ch);
        if(head==NULL)
        head=tail=newnode;
        else{
            tail->next=newnode;
            tail=newnode;
        }
    }
    scanf("%d",&index);
    scanf(" %c",&newchar);
    node*oldhead=head;
    insertAfter(head,index,newchar);
```

```
    printlist(oldhead);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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;
};

// You are using GCC
void insertAtFront(struct Node** head, int newdata ){
    struct Node* newnode=(struct Node*) malloc(sizeof(struct Node));
    newnode->data=newdata;
    newnode->next= *head;
    *head=newnode;
}
struct Node* reverse(struct Node*head){
    struct Node*prev=NULL;
    struct Node*current=head;
    struct Node*next=NULL;
    while(current !=NULL){
        next=current->next;
        current->next=prev;
        prev=current;
        current=next;
    }
    return prev;

}
```

```
void printList(struct Node*node){
    while(node != NULL){
        printf("%d",node->data);
        node=node->next;
    }
    printf("\n");
}

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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct Node{
    float gpa;
    struct Node*next;
}Node;
Node*insert(Node*head,float gpa){
    Node*newNode=(Node*)malloc(sizeof(Node));
    if(!newNode){
        printf("Memory allocation failed\n");
        return head;
    }
    newNode->gpa=gpa;
    newNode->next=head;
    return newNode;
}
Node*deleteNode(Node*head,int pos){
    if(!head)return head;
    Node*temp=head,*prev=NULL;
```

```
    if(pos==1){
        head=head->next;
        free(temp);
        return head;
    }
    for(int i=1;temp&&i<pos;i++){
        prev=temp;
        temp=temp->next;
    }
    if(!temp)return head;
    prev->next=temp->next;
    free(temp);
    return head;
}
void display(Node*head){
    while(head){
        printf("GPA: %.1f\n",head->gpa);
        head=head->next;
    }
}
int main(){
    int n,pos;
    float gpa;
    Node*head=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%f",&gpa);
        head=insert(head,gpa);
    }
    scanf("%d",&pos);
    head=deleteNode(head,pos);
    display(head);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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>
struct node{
   int data;
   struct node*next;
};
void insert(struct node**head,int data){
   struct node*newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=data;
   newnode->next=NULL;
   if(*head==NULL){
      *head=newnode;
      return;
   }
   struct node*temp=*head;
   while(temp->next !=NULL){
      temp=temp->next;
   }temp->next=newnode;
}
void display(struct node*head){
   struct node*temp=head;
   while(temp!=NULL){
      printf("%d ",temp->data);
      temp=temp->next;
   }printf("\n");
}
int  main(){
   int n,value;
```

```
    struct node*head=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&value);
        insert(&head,value);
    }display(head);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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(struct Node*head,int data){
    struct Node*newnode=(struct Node*)malloc(sizeof(struct Node));;
    newnode->data=data;
    newnode->next=head;
    return newnode;
}
int printMiddle(struct Node*head){
    struct Node*slow=head,*fast=head;
    while(fast!=NULL && fast->next!=NULL){
        slow=slow->next;
        fast=fast->next->next;
```

```
    }
    return slow->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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 19

Section 1 : MCQ

1. What is the main advantage of a two-way linked list over a one-way linked list?

Answer

Two-way linked lists allow for traversal in both directions.

Status : Correct ***Marks : 1/1***

2. What does the following code snippet do?

```
struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
newNode->data = value;
newNode->next = NULL;
newNode->prev = NULL;
```

Answer

Creates a new node and initializes its data to 'value'

Status : Correct **Marks : 1/1**

3. What will be the output of the following program?

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

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

int main() {
    struct Node* head = NULL;
    struct Node* tail = NULL;
    for (int i = 0; i < 5; i++) {
        struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
        temp->data = i + 1;
        temp->prev = tail;
        temp->next = NULL;
        if (tail != NULL) {
            tail->next = temp;
        } else {
            head = temp;
        }
        tail = temp;
    }
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    return 0;
}
```

Answer

1 2 3 4 5

Status : Correct ***Marks : 1/1***

4. Which code snippet correctly deletes a node with a given value from a doubly linked list?

```
void deleteNode(Node** head_ref, Node* del_node) {
    if (*head_ref == NULL || del_node == NULL) {
        return;
    }
    if (*head_ref == del_node) {
        *head_ref = del_node->next;
    }
    if (del_node->next != NULL) {
        del_node->next->prev = del_node->prev;
    }
    if (del_node->prev != NULL) {
        del_node->prev->next = del_node->next;
    }
    free(del_node);
}
```

Answer

Deletes the node at a given position in a doubly linked list.

Status : Wrong ***Marks : 0/1***

5. Consider the following function that refers to the head of a Doubly Linked List as the parameter. Assume that a node of a doubly linked list has the previous pointer as prev and the next pointer as next.

Assume that the reference of the head of the following doubly linked list is passed to the below function 1 <--> 2 <--> 3 <--> 4 <--> 5 <-->6. What should be the modified linked list after the function call?

```
Procedure fun(head_ref: Pointer to Pointer of node)
    temp = NULL
    current = *head_ref
```

```
    While current is not NULL
        temp = current->prev
        current->prev = current->next
        current->next = temp
        current = current->prev
    End While

    If temp is not NULL
        *head_ref = temp->prev
    End If
End Procedure
```

Answer

6 <--> 5 <--> 4 <--> 3 <--> 2 <--> 1.

Status : Correct ***Marks : 1/1***

6. What is the correct way to add a node at the beginning of a doubly linked list?

Answer

void addFirst(int data){ Node* newNode = new Node(data); newNode->next = head; if (head != NULL) { head->prev = newNode; } head = newNode; }

Status : Correct ***Marks : 1/1***

7. What will be the effect of setting the prev pointer of a node to NULL in a doubly linked list?

Answer

The node will become the new head

Status : Correct ***Marks : 1/1***

8. What is a memory-efficient double-linked list?

Answer

A doubly linked list that uses bitwise AND operator for storing addresses

Status : Correct ***Marks : 1/1***

9. What will be the output of the following code?

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

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

int main() {
    struct Node* head = NULL;
    struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
    temp->data = 2;
    temp->next = NULL;
    temp->prev = NULL;
    head = temp;
    printf("%d\n", head->data);
    free(temp);
    return 0;
}
```

Answer

2

Status : Correct ***Marks : 1/1***

10. Which of the following is true about the last node in a doubly linked list?

Answer

Its next pointer is NULL

Status : Correct ***Marks : 1/1***

11. How do you reverse a doubly linked list?

Answer

By swapping the next and previous pointers of each node

Status : Correct ***Marks : 1/1***

12. Which of the following is false about a doubly linked list?

Answer

Implementing a doubly linked list is easier than singly linked list

Status : Correct ***Marks : 1/1***

13. Which pointer helps in traversing a doubly linked list in reverse order?

Answer

prev

Status : Correct ***Marks : 1/1***

14. Which of the following statements correctly creates a new node for a doubly linked list?

Answer

struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));

Status : Correct ***Marks : 1/1***

15. Consider the provided pseudo code. How can you initialize an empty two-way linked list?

Define Structure Node
  data: Integer

prev: Pointer to Node
next: Pointer to Node
End Define

Define Structure TwoWayLinkedList
head: Pointer to Node
tail: Pointer to Node
End Define

Answer

struct TwoWayLinkedList* list = malloc(sizeof(struct TwoWayLinkedList)); list->head = NULL; list->tail = NULL;

Status : Correct ***Marks : 1/1***

16. How do you delete a node from the middle of a doubly linked list?

Answer

All of the mentioned options

Status : Correct ***Marks : 1/1***

17. Where Fwd and Bwd represent forward and backward links to the adjacent elements of the list. Which of the following segments of code deletes the node pointed to by X from the doubly linked list, if it is assumed that X points to neither the first nor the last node of the list?

A doubly linked list is declared as

```
struct Node {
    int Value;
    struct Node *Fwd;
    struct Node *Bwd;
);
```

Answer

X->Bwd->Fwd = X->Fwd; X->Fwd->Bwd = X->Bwd;

Status : Correct ***Marks : 1/1***

18. How many pointers does a node in a doubly linked list have?

Answer

2

Status : Correct ***Marks : 1/1***

19. What happens if we insert a node at the beginning of a doubly linked list?

Answer

The previous pointer of the new node is NULL

Status : Correct ***Marks : 1/1***

20. Which of the following information is stored in a doubly-linked list's nodes?

Answer

All of the mentioned options

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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;
};

class Node:
    def __init__(self, data):
        self.data = data
        self.prev = None
        self.next = None

class DoublyLinkedList:
    def __init__(self):
        self.head = None
        self.tail = None
```

```
    def insert_at_end(self, data):
        new_node = Node(data)
        if not self.head:  # If list is empty
            self.head = self.tail = new_node
        else:
            self.tail.next = new_node
            new_node.prev = self.tail
            self.tail = new_node

    def display_forward(self):
        current = self.head
        while current:
            print(current.data, end=' ')
            current = current.next

    def display_backward(self):
        current = self.tail
        while current:
            print(current.data, end=' ')
            current = current.prev

# Main Program
dll = DoublyLinkedList()

items = input().split()
for item in items:
    if item == '-':
        break
    dll.insert_at_end(item)

print("Forward Playlist: ", end='')
dll.display_forward()
print("Backward Playlist: ", end='')
dll.display_backward()
```

```
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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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>

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

Node* head = NULL;
Node* tail = NULL;

// Append a node to the end of the doubly linked list
void append(int id) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->id = id;
    newNode->next = NULL;
    newNode->prev = tail;

    if (tail != NULL) {
        tail->next = newNode;
    } else {
        head = newNode; // first node
    }
```

```
    tail = newNode;
}

// Print the maximum ID in the list
void printMaxID() {
    if (head == NULL) {
        printf("Empty list!");
        return;
    }

    int maxID = head->id;
    Node* current = head->next;
    while (current != NULL) {
        if (current->id > maxID) {
            maxID = current->id;
        }
        current = current->next;
    }
    printf("%d", maxID);
}

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

    if (n == 0) {
        printf("Empty list!");
        return 0;
    }

    for (int i = 0; i < n; i++) {
        int id;
        scanf("%d", &id);
        append(id);
    }

    printMaxID();

    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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;
```

```
# You are using Python
class Node:
    def __init__(self, data):
        self.data = data
        self.prev = None
        self.next = None

class DoublyLinkedList:
    def __init__(self):
```

```
        self.head = None

    def insert_front(self, data):
        new_node = Node(data)
        if self.head is not None:
            self.head.prev = new_node
            new_node.next = self.head
        self.head = new_node
        print("Node Inserted")
        self.display()

    def display(self):
        temp = self.head
        while temp:
            print(temp.data, end=' ')
            temp = temp.next
        print()

# Main program
n = int(input())
ids = list(map(int, input().split()))

dll = DoublyLinkedList()

for emp_id in ids:
    dll.insert_front(emp_id)
```

```
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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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>

// Structure for a node in the doubly linked list
typedef struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
} Node;

// Structure for the doubly linked list
typedef struct {
    Node* head;
} DoublyLinkedList;

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory error\n");
        return NULL;
    }
    newNode->data = data;
    newNode->prev = NULL;
```

```
    newNode->next = NULL;
    return newNode;
}

// Function to create a doubly linked list and insert student IDs
void createList(DoublyLinkedList* list, int N, int* student_ids) {
    for (int i = 0; i < N; i++) {
        Node* newNode = createNode(student_ids[i]);
        if (list->head == NULL) {
            list->head = newNode;
        } else {
            Node* current = list->head;
            while (current->next) {
                current = current->next;
            }
            current->next = newNode;
            newNode->prev = current;
        }
    }
}

// Function to display the student IDs stored in the doubly linked list
void displayList(DoublyLinkedList* list) {
    Node* current = list->head;
    while (current) {
        printf("%d ", current->data);
        current = current->next;
    }
}

int main() {
    // Read the number of student IDs
    int N;
    scanf("%d", &N);

    // Read the student IDs
    int student_ids[N];
    for (int i = 0; i < N; i++) {
        scanf("%d", &student_ids[i]);
    }

    // Create a doubly linked list and insert student IDs
```

```
    DoublyLinkedList list;
    list.head = NULL;
    createList(&list, N, student_ids);

    // Display the student IDs stored in the doubly linked list
    displayList(&list);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

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

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

// Insert node at the end
void insertEnd(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
    newNode->prev = temp;
}

// Display the list
void displayList(Node* head) {
    int index = 1;
    while (head != NULL) {
        printf(" node %d : %d\n", index++, head->data);
        head = head->next;
    }
}

// Delete node at given position
int deleteAtPosition(Node** head, int position, int n) {
    if (position < 1 || position > n)
        return 0;

    Node* temp = *head;
```

```
    // Deleting head
    if (position == 1) {
        *head = temp->next;
        if (*head != NULL)
            (*head)->prev = NULL;
        free(temp);
        return 1;
    }

    for (int i = 1; i < position; i++) {
        temp = temp->next;
    }

    if (temp == NULL)
        return 0;

    if (temp->prev != NULL)
        temp->prev->next = temp->next;

    if (temp->next != NULL)
        temp->next->prev = temp->prev;

    free(temp);
    return 1;
}

int main() {
    int n, i, value, pos;
    Node* head = NULL;

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

    scanf("%d", &pos);

    printf("Data entered in the list:\n");
    displayList(head);
```

```
    if (pos < 1 || pos > n) {
        printf("Invalid position. Try again.");
        return 0;
    }

    deleteAtPosition(&head, pos, n);

    printf("\nAfter deletion the new list:\n");
    displayList(head);

    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 19

Section 1 : MCQ

1. Here is an Infix Expression: 4+3*(6*3-12). Convert the expression from Infix to Postfix notation. The maximum number of symbols that will appear on the stack AT ONE TIME during the conversion of this expression?

Answer

3

Status : Wrong ***Marks : 0/1***

2. In a stack data structure, what is the fundamental rule that is followed for performing operations?

Answer

Last In First Out

Status : Correct ***Marks : 1/1***

3. Consider the linked list implementation of a stack.

Which of the following nodes is considered as Top of the stack?

Answer

First node

Status : Correct ***Marks : 1/1***

4. The user performs the following operations on the stack of size 5 then at the end of the last operation, the total number of elements present in the stack is

```
push(1);
pop();
push(2);
push(3);
pop();
push(4);
pop();
pop();
push(5);
```

Answer

1

Status : Correct ***Marks : 1/1***

5. In an array-based stack, which of the following operations can result in a Stack underflow?

Answer

Popping an element from an empty stack

Status : Correct ***Marks : 1/1***

6. A user performs the following operations on stack of size 5 then which of the following is correct statement for Stack?

```
push(1);
pop();
push(2);
push(3);
pop();
push(2);
pop();
pop();
push(4);
pop();
pop();
push(5);
```

Answer

Underflow Occurs

Status : Correct ***Marks : 1/1***

7. What will be the output of the following code?

```
#include <stdio.h>
#define MAX_SIZE 5
int stack[MAX_SIZE];
int top = -1;
int isEmpty() {
   return (top == -1);
}
int isFull() {
   return (top == MAX_SIZE - 1);
}
void push(int item) {
   if (isFull())
      printf("Stack Overflow\n");
   else
      stack[++top] = item;
}
```

```
int main() {
    printf("%d\n", isEmpty());
    push(10);
    push(20);
    push(30);
    printf("%d\n", isFull());
    return 0;
}
```

Answer

10

Status : Correct ***Marks : 1/1***

8. What is the advantage of using a linked list over an array for implementing a stack?

Answer

Linked lists can dynamically resize

Status : Correct ***Marks : 1/1***

9. The result after evaluating the postfix expression 10 5 + 60 6 / * 8 - is

Answer

142

Status : Correct ***Marks : 1/1***

10. What will be the output of the following code?

```
#include <stdio.h>
#define MAX_SIZE 5
int stack[MAX_SIZE];
int top = -1;
void display() {
    if (top == -1) {
        printf("Stack is empty\n");
```

```
    } else {
        printf("Stack elements: ");
        for (int i = top; i >= 0; i--) {
            printf("%d ", stack[i]);
        }
        printf("\n");
    }
}
void push(int value) {
    if (top == MAX_SIZE - 1) {
        printf("Stack Overflow\n");
    } else {
        stack[++top] = value;
    }
}
int main() {
    display();
    push(10);
    push(20);
    push(30);
    display();
    push(40);
    push(50);
    push(60);
    display();
    return 0;
}
```

Answer

Stack is emptyStack elements: 30 20 10Stack OverflowStack elements: 50 40 30 20 10

Status : Correct ***Marks : 1/1***

11. In the linked list implementation of the stack, which of the following operations removes an element from the top?

Answer

Pop

Status : Correct **Marks : 1/1**

12. What is the primary advantage of using an array-based stack with a fixed size?

Answer

Efficient memory usage

Status : Correct **Marks : 1/1**

13. Consider a linked list implementation of stack data structure with three operations:

push(value): Pushes an element value onto the stack.pop(): Pops the top element from the stack.top(): Returns the item stored at the top of the stack.

Given the following sequence of operations:

push(10);pop();push(5);top();

What will be the result of the stack after performing these operations?

Answer

The top element in the stack is 5

Status : Correct **Marks : 1/1**

14. Which of the following Applications may use a Stack?

Answer

All of the mentioned options

Status : Correct **Marks : 1/1**

15. When you push an element onto a linked list-based stack, where does the new element get added?

Answer

At the beginning of the list

Status : Correct **Marks : 1/1**

16. What is the value of the postfix expression 6 3 2 4 + - *?

Answer

-18

Status : Correct **Marks : 1/1**

17. What will be the output of the following code?

```
#include <stdio.h>
#define MAX_SIZE 5
void push(int* stack, int* top, int item) {
    if (*top == MAX_SIZE - 1) {
        printf("Stack Overflow\n");
        return;
    }
    stack[++(*top)] = item;
}
int pop(int* stack, int* top) {
    if (*top == -1) {
        printf("Stack Underflow\n");
        return -1;
    }
    return stack[(*top)--];
}

int main() {
    int stack[MAX_SIZE];
    int top = -1;
    push(stack, &top, 10);
    push(stack, &top, 20);
    push(stack, &top, 30);
    printf("%d\n", pop(stack, &top));
    printf("%d\n", pop(stack, &top));
```

```
    printf("%d\n", pop(stack, &top));
    printf("%d\n", pop(stack, &top));
    return 0;
}
```

Answer

302010Stack Underflow-1

Status : Correct ***Marks : 1/1***

18. Which of the following operations allows you to examine the top element of a stack without removing it?

Answer

Peek

Status : Correct ***Marks : 1/1***

19. Elements are Added on _____ of the Stack.

Answer

Top

Status : Correct ***Marks : 1/1***

20. Pushing an element into the stack already has five elements. The stack size is 5, then the stack becomes

Answer

Overflow

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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 ", value);
}void pop() {
    if (top == NULL) {
        printf("Stack is empty. Cannot pop. ");
```

```
    } else {
        struct Node* temp = top;
        printf("Popped element: %d ", temp->data);
        top = top->next;
        free(temp);
    }
}void displayStack() {
    if (top == NULL) {
        printf("Stack is empty ");
    } else {
        struct Node* current = top;
        printf("Stack elements (top to bottom): ");
        while (current != NULL) {
            printf("%d", current->data);
            if (current->next != NULL) printf(" ");
            current = current->next;
        }
        printf(" ");
    }
}

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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

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

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

struct Node* top = NULL;

void push(int bookID) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        // Normally you would handle memory overflow, but not needed as per
question
        return;
    }
    newNode->bookID = bookID;
    newNode->next = top;
    top = newNode;
    printf("Book ID %d is pushed onto the stack ", bookID);
}

void pop() {
    if (top == NULL) {
        printf("Stack Underflow ");
    } else {
        struct Node* temp = top;
        printf("Book ID %d is popped from the stack ", top->bookID);
        top = top->next;
        free(temp);
    }
}

void display() {
    if (top == NULL) {
```

```
        printf("Stack is empty ");
    } else {
        struct Node* temp = top;
        printf("Book ID in the stack:");
        while (temp != NULL) {
            printf(" %d", temp->bookID);
            temp = temp->next;
        }
        printf(" ");
    }
}

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");
                return 0;
            default:
                printf("Invalid choice ");
                break;
        }
    }
    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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 ch) {
   if (top < MAX_SIZE - 1) {
      items[++top] = ch;
      printf("Pushed: %c ", ch);
   }
}

void pop() {
   if (top == -1) {
      printf("Stack is empty. Nothing to pop. ");
   } else {
      printf("Popped: %c ", items[top--]);
   }
}

void display() {
   if (top == -1) {
      printf("Stack is empty. ");
   } else {
      printf("Stack elements:");
      for (int i = top; i >= 0; i--) {
         printf(" %c", items[i]);
      }
```

```
        printf(" ");
    }
}

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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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') ||
           (ch >= '0' && ch <= '9');
}

int Prec(char ch) {
    switch (ch) {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
```

```
        return 2;
    case '^':
        return 3;
    }
    return -1;
}

// The corrected infixToPostfix function
void infixToPostfix(char* exp) {
    int i, k = 0;
    int len = strlen(exp);
    struct Stack* stack = createStack(len);
    if (!stack) {
        printf("Stack creation failed!\n");
        return;
    }

    char* output = (char*)malloc((len + 1) * sizeof(char)); // Output array
    if (!output) {
        printf("Memory allocation for output failed!\n");
        free(stack->array);
        free(stack);
        return;
    }

    for (i = 0; exp[i]; i++) {
        if (isOperand(exp[i])) {
            output[k++] = exp[i];
        }
        else if (exp[i] == '(') {
            push(stack, exp[i]);
        }
        else if (exp[i] == ')') {
            while (!isEmpty(stack) && peek(stack) != '(')
                output[k++] = pop(stack);
            if (!isEmpty(stack) && peek(stack) == '(')
                pop(stack);  // pop the '('
            else {
                printf("Invalid Expression (mismatched parentheses)\n");
                free(stack->array);
                free(stack);
                free(output);
```

```
            return;
        }
    }
    else { // Operator
        while (!isEmpty(stack) && Prec(exp[i]) <= Prec(peek(stack))) {
            if (exp[i] == '^' && peek(stack) == '^')
                break; // Right associativity of ^
            else
                output[k++] = pop(stack);
        }
        push(stack, exp[i]);
    }
}

// Pop remaining elements
while (!isEmpty(stack))
    output[k++] = pop(stack);

output[k] = '\0'; // Null-terminate the output

printf("%s\n", output);

free(stack->array);
free(stack);
free(output);
}

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

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

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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 ch) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = ch;
    newNode->next = top;
    top = newNode;
    printf("Adding Section: %c\n", ch);
}

// Pop operation with no arguments
void pop() {
    if (top == NULL) {
        printf("Stack is empty. Cannot pop.\n");
        return;
    }
    Node* temp = top;
    char ch = temp->data;
    top = top->next;
    free(temp);
    printf("Removing Section: %c\n", ch);
}
```

```
// Display operation
void displayStack() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return;
    }
    printf("Enrolled Sections: ");
    Node* current = top;
    while (current != NULL) {
        printf("%c ", current->data);
        current = current->next;
    }
    printf("\n");
}

// Free all nodes before exiting
void freeStack() {
    while (top != NULL) {
        Node* temp = top;
        top = temp->next;
        free(temp);
    }
}


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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 16

Section 1 : MCQ

1. In a linked list implementation of a queue, front and rear pointers are tracked. Which of these pointers will change during an insertion into a non-empty queue?

Answer

Only rear pointer

Status : Correct ***Marks : 1/1***

2. When new data has to be inserted into a stack or queue, but there is no available space. This is known as

Answer

overflow

Status : Correct *Marks : 1/1*

3. What will be the output of the following code?

```
#include <stdio.h>
#define MAX_SIZE 5
typedef struct {
    int arr[MAX_SIZE];
    int front;
    int rear;
    int size;
} Queue;

void enqueue(Queue* queue, int data) {
    if (queue->size == MAX_SIZE) {
        return;
    }
    queue->rear = (queue->rear + 1) % MAX_SIZE;
    queue->arr[queue->rear] = data;
    queue->size++;
}
int dequeue(Queue* queue) {
    if (queue->size == 0) {
        return -1;
    }
    int data = queue->arr[queue->front];
    queue->front = (queue->front + 1) % MAX_SIZE;
    queue->size--;
    return data;
}
int main() {
    Queue queue;
    queue.front = 0;
    queue.rear = -1;
    queue.size = 0;
    enqueue(&queue, 1);
    enqueue(&queue, 2);
    enqueue(&queue, 3);
```

```
    printf("%d ", dequeue(&queue));
    printf("%d ", dequeue(&queue));
    enqueue(&queue, 4);
    enqueue(&queue, 5);
    printf("%d ", dequeue(&queue));
    printf("%d ", dequeue(&queue));
    return 0;
}
```

Answer

1 2 3 4

Status : Correct ***Marks : 1/1***

4. What will be the output of the following code?

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 5
typedef struct {
    int* arr;
    int front;
    int rear;
    int size;
} Queue;
Queue* createQueue() {
    Queue* queue = (Queue*)malloc(sizeof(Queue));
    queue->arr = (int*)malloc(MAX_SIZE * sizeof(int));
    queue->front = -1;
    queue->rear = -1;
    queue->size = 0;
    return queue;
}
int isEmpty(Queue* queue) {
    return (queue->size == 0);
}
int main() {
    Queue* queue = createQueue();
    printf("Is the queue empty? %d", isEmpty(queue));
```

```
    return 0;
}
```

Answer

Is the queue empty? 1

Status : Correct ***Marks : 1/1***

5. The process of accessing data stored in a serial access memory is similar to manipulating data on a

Answer

Stack

Status : Wrong ***Marks : 0/1***

6. Which of the following can be used to delete an element from the front end of the queue?

Answer

public Object deleteFront() throws emptyDEQException{if(isEmpty())throw new emptyDEQException("Empty");else{Node temp = head.getNext();Node cur = temp.getNext();Object e = temp.getEle();head.setNext(temp);size--;return e;}}

Status : Wrong ***Marks : 0/1***

7. What does the front pointer in a linked list implementation of a queue contain?

Answer

The address of the first element

Status : Correct ***Marks : 1/1***

8. Insertion and deletion operation in the queue is known as

Answer

Enqueue and Dequeue

Status : Correct ***Marks : 1/1***

9. Which of the following properties is associated with a queue?

Answer

First In First Out

Status : Correct ***Marks : 1/1***

10. A normal queue, if implemented using an array of size MAX_SIZE, gets full when

Answer

Front = (rear + 1)mod MAX_SIZE

Status : Wrong ***Marks : 0/1***

11. In what order will they be removed If the elements “A”, “B”, “C” and “D” are placed in a queue and are deleted one at a time

Answer

ABCD

Status : Correct ***Marks : 1/1***

12. What will the output of the following code?

```
#include <stdio.h>
#include <stdlib.h>
typedef struct {
    int* arr;
    int front;
    int rear;
    int size;
} Queue;
```

```
Queue* createQueue() {
    Queue* queue = (Queue*)malloc(sizeof(Queue));
    queue->arr = (int*)malloc(5 * sizeof(int));
    queue->front = 0;
    queue->rear = -1;
    queue->size = 0;
    return queue;
}
int main() {
    Queue* queue = createQueue();
    printf("%d", queue->size);
    return 0;
}
```

Answer

0

Status : Correct ***Marks : 1/1***

13. The essential condition that is checked before insertion in a queue is?

Answer

Overflow

Status : Correct ***Marks : 1/1***

14. In linked list implementation of a queue, the important condition for a queue to be empty is?

Answer

FRONT is null

Status : Correct ***Marks : 1/1***

15. After performing this set of operations, what does the final list look to contain?

InsertFront(10);

InsertFront(20);
InsertRear(30);
DeleteFront();
InsertRear(40);
InsertRear(10);
DeleteRear();
InsertRear(15);
display();

Answer

20 30 40 15

Status : Wrong ***Marks : 0/1***

16. Front and rear pointers are tracked in the linked list implementation of a queue. Which of these pointers will change during an insertion into the EMPTY queue?

Answer

Both front and rear pointer

Status : Correct ***Marks : 1/1***

17. What are the applications of dequeue?

Answer

All the mentioned options

Status : Correct ***Marks : 1/1***

18. What is the functionality of the following piece of code?

```
public void function(Object item)
{
   Node temp=new Node(item,trail);
   if(isEmpty())
   {
      head.setNext(temp);
```

```
        temp.setNext(trail);
    }
    else
    {
        Node cur=head.getNext();
        while(cur.getNext()!=trail)
        {
            cur=cur.getNext();
        }
        cur.setNext(temp);
    }
    size++;
}
```

Answer

Insert at the rear end of the dequeue

Status : Correct ***Marks : 1/1***

19. Which one of the following is an application of Queue Data Structure?

Answer

All of the mentioned options

Status : Correct ***Marks : 1/1***

20. Which operations are performed when deleting an element from an array-based queue?

Answer

Dequeue

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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;
}
MAX = 5
```

```
queue = []
front = 0
rear = 0

import sys
inputs = sys.stdin.read().split()
i = 0

while i < len(inputs):
    choice = inputs[i]

    if choice == '1':
        i += 1
        if i < len(inputs):
            order = inputs[i]
            if rear - front < MAX:
                queue.append(order)
                rear += 1
                print(f"Order for {order} is enqueued.", end=" ")
            else:
                print("Queue is full. Cannot enqueue more orders.", end=" ")
        else:
            print("Invalid option.", end=" ")
    elif choice == '2':
        if rear == front:
            print("No orders in the queue.", end=" ")
        else:
            print(f"Dequeued Order: {queue[front]}", end=" ")
            front += 1
    elif choice == '3':
        if rear == front:
            print("Queue is empty. No orders available.", end=" ")
        else:
            print("Orders in the queue are:", end=" ")
            for j in range(front, rear):
                print(queue[j], end=" ")
    elif choice == '4':
        print("Exiting program", end=" ")
        break
    else:
        print("Invalid option.", end=" ")
    i += 1
```

```
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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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 || front > rear);
}

int isFull() {
    return (rear == MAX_SIZE - 1);
}

void enqueue(int ticketID) {
    if (isFull()) {
        printf("Queue is full. Cannot enqueue.\n");
        return;
    }
    if (front == -1) front = 0;
    rear++;
    ticketIDs[rear] = ticketID;
    printf("Helpdesk Ticket ID %d is enqueued.\n", ticketID);
}

int dequeue() {
    if (isEmpty()) {
        printf("Queue is empty.\n");
        return 0;
    }
    else{
    lastDequeued = ticketIDs[front];
    front++;
    if (front > rear) {
        front = rear = -1;
    }
    return 1;}
}

void display() {
    if (isEmpty()) {
        printf("Queue is empty.\n");
        return;
    }
    printf("Helpdesk Ticket IDs in the queue are: ");
    for (int i = front; i <= rear; i++) {
```

```
        printf("%d", ticketIDs[i]);
        if (i != rear) printf(" ");
    }
    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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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;
```

```
MAX = 5

queue = []
front = 0  # will track the index of the front element

while True:
    try:
        inputs = input().strip().split()
        if len(inputs) == 0:
            continue
        choice = int(inputs[0])

        if choice == 1:
            # Insert operation
            if len(inputs) != 2:
                print("Invalid option.")
                continue
            data = int(inputs[1])
            if len(queue) == MAX:
                print("Queue is full.")
            else:
                queue.append(data)
                print(f"{data} is inserted in the queue.")

        elif choice == 2:
            # Delete operation
            if len(queue) == 0:
```

```
            print("Queue is empty.")
        else:
            deleted = queue.pop(0)
            print(f"Deleted number is: {deleted}")

    elif choice == 3:
        # Display operation
        if len(queue) == 0:
            print("Queue is empty.")
        else:
            print("Elements in the queue are: " + " ".join(map(str, queue)) + " ")

    else:
        # Invalid option
        print("Invalid option.")

  except EOFError:
    break
  except:
    # Any other unexpected error in input format
    print("Invalid option.")
```

```
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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 4

Attempt : 1
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

```
int isFull() {
    return rear == MAX_SIZE;
}

int isEmpty() {
    return front == rear;
```

```
}

void enqueue(int pages) {
    if (isFull()) {
        printf("Queue is full. Cannot enqueue.\n");
        return;
    }
    queue[rear] = pages;
    rear++;
    printf("Print job with %d pages is enqueued.\n", pages);
}

void dequeue() {
    if (isEmpty()) {
        printf("Queue is empty.\n");
        return;
    }
    printf("Processing print job: %d pages\n", queue[front]);
    front++;
    // Shift elements to the left to keep front at 0
    if (front == rear) {
        // Queue is empty now, reset pointers
        front = rear = 0;
    } else {
        for (int i = front; i < rear; i++) {
            queue[i - front] = queue[i];
        }
        rear = rear - front;
        front = 0;
    }
}

void display() {
    if (isEmpty()) {
        printf("Queue is empty.\n");
        return;
    }
    printf("Print jobs in the queue: ");
    for (int i = front; i < rear; i++) {
        printf("%d", queue[i]);
        if (i != rear - 1) printf(" ");
    }
```

```
    printf(" \n");
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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;
```

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
```

```
class Queue:
    def __init__(self):
        self.front = None
        self.rear = None

    def enqueue(self, data):
        new_node = Node(data)
        if self.rear is None:
            self.front = self.rear = new_node
        else:
            self.rear.next = new_node
            self.rear = new_node

    def dequeue(self):
        if self.front is None:
            return None
        removed_data = self.front.data
        self.front = self.front.next
        if self.front is None:
            self.rear = None
        return removed_data

    def get_front_rear(self):
        if self.front is None:
            return None, None
        return self.front.data, self.rear.data

def main():
    n = int(input())
    elements = list(map(int, input().split()))

    q = Queue()
    for elem in elements:
        q.enqueue(elem)

    front, rear = q.get_front_rear()
    print(f"Front: {front}, Rear: {rear}")
    print("Performing Dequeue Operation:")
    q.dequeue()
    front, rear = q.get_front_rear()
    print(f"Front: {front}, Rear: {rear}")
```

```
if __name__ == "__main__":
    main()
```

```
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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 15

Section 1 : MCQ

1. Which of the following is the correct pre-order traversal of a binary search tree with nodes: 50, 30, 20, 55, 32, 52, 57?

Answer

50, 30, 20, 32, 55, 52, 57

Status : Correct ***Marks : 1/1***

2. Find the postorder traversal of the given binary search tree.

Answer

1, 4, 2, 18, 14, 13

Status : Correct ***Marks : 1/1***

3. Which of the following is the correct in-order traversal of a binary search tree with nodes: 9, 3, 5, 11, 8, 4, 2?

Answer

2, 3, 4, 5, 8, 9, 11

Status : Correct ***Marks : 1/1***

4. Find the in-order traversal of the given binary search tree.

Answer

1, 2, 4, 13, 14, 18

Status : Correct ***Marks : 1/1***

5. The preorder traversal of a binary search tree is 15, 10, 12, 11, 20, 18, 16, 19. Which one of the following is the postorder traversal of the tree?

Answer

11, 12, 10, 16, 19, 18, 20, 15

Status : Correct ***Marks : 1/1***

6. In a binary search tree with nodes 18, 28, 12, 11, 16, 14, 17, what is the value of the left child of the node 16?

Answer

14

Status : Correct ***Marks : 1/1***

7. While inserting the elements 5, 4, 2, 8, 7, 10, 12 in a binary search tree, the element at the lowest level is _____.

Answer

12

Status : Correct ***Marks : 1/1***

8. How many distinct binary search trees can be created out of 4 distinct keys?

Answer

14

Status : Correct ***Marks : 1/1***

9. Find the pre-order traversal of the given binary search tree.

Answer

13, 2, 1, 4, 14, 18

Status : Correct ***Marks : 1/1***

10. Which of the following is the correct post-order traversal of a binary search tree with nodes: 50, 30, 20, 55, 32, 52, 57?

Answer

20, 32, 30, 52, 57, 55, 50

Status : Correct ***Marks : 1/1***

11. Which of the following operations can be used to traverse a Binary Search Tree (BST) in ascending order?

Answer

Inorder traversal

Status : Correct ***Marks : 1/1***

12. Find the preorder traversal of the given binary search tree.

Answer

9, 2, 1, 6, 4, 7, 10, 14

Status : Correct ***Marks : 1/1***

13. While inserting the elements 71, 65, 84, 69, 67, 83 in an empty binary search tree (BST) in the sequence shown, the element in the lowest level is _____.

Answer

67

Status : Correct ***Marks : 1/1***

14. Which of the following is a valid preorder traversal of the binary search tree with nodes: 18, 28, 12, 11, 16, 14, 17?

Answer

18, 12, 11, 16, 14, 17, 28

Status : Correct ***Marks : 1/1***

15. Find the post-order traversal of the given binary search tree.

Answer

10, 17, 20, 18, 15, 32, 21

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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) {
    if(root==NULL){
```

```
        return  createNode(key);
    }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) {
    while(root && root->left!=NULL){
        root=root->left;
    }return root;
}

struct TreeNode* deleteNode(struct TreeNode* root, int key) {
    if(root==NULL) return NULL;

    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==NULL){
            struct TreeNode*temp=root->right;
            free(root);
            return temp;
        }else if(root->right==NULL){
            struct TreeNode*temp=root->left;
            free(root);
            return temp;
        }struct TreeNode*temp=findMin(root->right);
        root->data=temp->data;
        root->right=deleteNode(root->right,temp->data);
    }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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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) {if(root==NULL){return
createNode(value);}if(value < root->data){root->left=insert(root->left,value);}
else{root->right=insert(root->right,value);}return root;}void printPreorder(struct
Node* node) {if(node==NULL) return;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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

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

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

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

Node* insert(Node* root, int key) {
    if (root == NULL)
        return newNode(key);
    if (key < root->key)
```

```
        root->left = insert(root->left, key);
    else
        root->right = insert(root->right, key);
    return root;
}

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

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

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

    scanf("%d", &key);

    if (search(root, key))
        printf("Value %d is found in the tree.\n", key);
    else
        printf("Value %d is not found in the tree.\n", key);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 4

Attempt : 1
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;
}

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

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

void displayTreePostOrder(struct Node* root) {
    if (root == NULL)
        return;
    displayTreePostOrder(root->left);
    displayTreePostOrder(root->right);
    printf("%d ", root->data);
}

int findMinValue(struct Node* root) {
    struct 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: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

```
}
int findMax(struct TreeNode* root) {
    struct TreeNode* current = root;
    while (current->right != NULL)
        current = current->right;
    return current->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**

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_MCQ_Updated_1

Attempt : 1
Total Mark : 20
Marks Obtained : 19

Section 1 : MCQ

1. Which of the following statements is true about the merge sort algorithm?

Answer

It requires additional memory for merging

Status : Correct ***Marks : 1/1***

2. What happens when Merge Sort is applied to a single-element array?

Answer

The array remains unchanged and no merging is required

Status : Correct ***Marks : 1/1***

3. Consider the Quick Sort algorithm, which sorts elements in ascending order using the first element as a pivot. Then which of the following input sequences will require the maximum number of comparisons when this algorithm is applied to it?

Answer

22 25 56 67 89

Status : Correct ***Marks : 1/1***

4. In a quick sort algorithm, where are smaller elements placed to the pivot during the partition process, assuming we are sorting in increasing order?

Answer

To the left of the pivot

Status : Correct ***Marks : 1/1***

5. Why is Merge Sort preferred for sorting large datasets compared to Quick Sort?

Answer

Merge Sort has better worst-case time complexity

Status : Correct ***Marks : 1/1***

6. Which of the following is not true about QuickSort?

Answer

It as an adaptive sorting algorithm

Status : Wrong ***Marks : 0/1***

7. Merge sort is _____.

Answer

Comparison-based sorting algorithm

Status : Correct **Marks : 1/1**

8. The following code snippet is an example of a quick sort. What do the 'low' and 'high' parameters represent in this code?

```
void quickSort(int arr[], int low, int high) {
   if (low < high) {
      int pivot = partition(arr, low, high);
      quickSort(arr, low, pivot - 1);
      quickSort(arr, pivot + 1, high);
   }
}
```

Answer

The range of elements to sort within the array

Status : Correct **Marks : 1/1**

9. What is the main advantage of Quicksort over Merge Sort?

Answer

Quicksort requires less auxiliary space

Status : Correct **Marks : 1/1**

10. Let P be a quick sort program to sort numbers in ascending order using the first element as a pivot. Let t1 and t2 be the number of comparisons made by P for the inputs {1, 2, 3, 4, 5} and {4, 1, 5, 3, 2}, respectively. Which one of the following holds?

Answer

t1 > t2

Status : Correct **Marks : 1/1**

11. Which of the following sorting algorithms is based on the divide and

conquer method?

Answer

Merge Sort

Status : Correct ***Marks : 1/1***

12. Is Merge Sort a stable sorting algorithm?

Answer

Yes, always stable.

Status : Correct ***Marks : 1/1***

13. Which of the following modifications can help Quicksort perform better on small subarrays?

Answer

Switching to Insertion Sort for small subarrays

Status : Correct ***Marks : 1/1***

14. Which of the following methods is used for sorting in merge sort?

Answer

merging

Status : Correct ***Marks : 1/1***

15. What is the best sorting algorithm to use for the elements in an array that are more than 1 million in general?

Answer

Quick sort.

Status : Correct ***Marks : 1/1***

16. Which of the following strategies is used to improve the efficiency of Quicksort in practical implementations?

Answer

Choosing the pivot randomly or using the median-of-three method

Status : Correct ***Marks : 1/1***

17. Which of the following is true about Quicksort?

Answer

It is an in-place sorting algorithm

Status : Correct ***Marks : 1/1***

18. Which of the following scenarios is Merge Sort preferred over Quick Sort?

Answer

When sorting linked lists

Status : Correct ***Marks : 1/1***

19. In a quick sort algorithm, what role does the pivot element play?

Answer

It is used to partition the array

Status : Correct ***Marks : 1/1***

20. What happens during the merge step in Merge Sort?

Answer

Two sorted subarrays are combined into one sorted array

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John and Mary are collaborating on a project that involves data analysis. They each have a set of age data, one sorted in ascending order and the other in descending order. However, their analysis requires the data to be in ascending order.

Write a program to help them merge the two sets of age data into a single sorted array in ascending order using merge sort.

Input Format

The first line of input consists of an integer N, representing the number of age values in each dataset.

The second line consists of N space-separated integers, representing the ages of participants in John's dataset (in ascending order).

The third line consists of N space-separated integers, representing the ages of participants in Mary's dataset (in descending order).

Output Format

The output prints a single line containing space-separated integers, which represents the merged dataset of ages sorted in ascending order.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
1 3 5 7 9
10 8 6 4 2

Output: 1 2 3 4 5 6 7 8 9 10

Answer

```
#include <stdio.h>


void merge(int arr[], int left[], int right[], int left_size, int right_size) {
    int i = 0, j = 0, k = 0;
    while (i < left_size && j < right_size) {
        if (left[i] <= right[j])
            arr[k++] = left[i++];
        else
            arr[k++] = right[j++];
    }
    while (i < left_size)
        arr[k++] = left[i++];
    while (j < right_size)
        arr[k++] = right[j++];
}

void mergeSort(int arr[], int size) {
    if (size < 2)
        return;
```

```
    int mid = size / 2;
    int left[mid];
    int right[size - mid];
 for (int i = 0; i < mid; i++)
        left[i] = arr[i];
    for (int i = mid; i < size; i++)
        right[i - mid] = arr[i];

    mergeSort(left, mid);
    mergeSort(right, size - mid);

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


int main() {
    int n, m;
    scanf("%d", &n);
    int arr1[n], arr2[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr1[i]);
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr2[i]);
    }
    int merged[n + n];
    mergeSort(arr1, n);
    mergeSort(arr2, n);
    merge(merged, arr1, arr2, n, n);
    for (int i = 0; i < n + n; i++) {
        printf("%d ", merged[i]);
    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Nandhini asked her students to arrange a set of numbers in ascending order. She asked the students to arrange the elements using insertion sort, which involves taking each element and placing it in its appropriate position within the sorted portion of the array.

Assist them in the task.

Input Format

The first line of input consists of the value of n, representing the number of array elements.

The second line consists of n elements, separated by a space.

Output Format

The output prints the sorted array, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
67 28 92 37 59

Output: 28 37 59 67 92

Answer

```
#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;
    }
}

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

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

```
    insertionSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are the lead developer of a text-processing application that assists writers in organizing their thoughts. One crucial feature is a character-sorting service that helps users highlight the most critical elements of their text.

To achieve this, you decide to enhance the service to sort characters in descending order using the Quick-Sort algorithm. Implement the algorithm to efficiently rearrange the characters, ensuring that it is sorted in descending order.

Input Format

The first line of the input consists of a positive integer value N, representing the number of characters to be sorted.

The second line of input consists of N space-separated lowercase alphabetical characters.

Output Format

The output displays the set of alphabetical characters, sorted in descending order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
a d g j k
Output: k j g d a

Answer

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

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

int partition(char arr[], int low, int high) {
    char 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]);
    return i + 1;
}

void quicksort(char arr[], int low, int high) {
```

```
    if (low < high) {
        int pi = partition(arr, low, high);
        quicksort(arr, low, pi - 1);
        quicksort(arr, pi + 1, high);
    }
}

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

    char characters[n];

    for (int i = 0; i < n; i++) {
        char input;
        scanf(" %c", &input);
        characters[i] = input;
    }

    quicksort(characters, 0, n - 1);

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

    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Kavya, a software developer, is analyzing data trends. She has a list of integers and wants to identify the nth largest number in the list after sorting the array using QuickSort.

To optimize performance, Kavya is required to use QuickSort to sort the list before finding the nth largest number.

Input Format

The first line of input consists of an integer n, representing the size of the array.

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

The third line consists of an integer k, representing the position of the largest

number you need to print after sorting the array.

Output Format

The output prints the k-th largest number in the sorted array (sorted in ascending order).

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
-1 0 1 2 -1 -4
3
Output: 0

Answer

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

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;
    for (int j = low; j < high; j++) {
        if (arr[j] <= pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return i + 1;
}

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
```

```
        quickSort(arr, pi + 1, high);
    }
}

void findNthLargest(int* nums, int n, int k) {
    quickSort(nums, 0, n - 1);
    printf("%d", nums[n - k]);
}

int main() {
    int n, k;
    scanf("%d", &n);
    int* nums = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }
    scanf("%d", &k);
    findNthLargest(nums, n, k);
    free(nums);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Jose has an array of N fractional values, represented as double-point numbers. He needs to sort these fractions in increasing order and seeks your help.

Write a program to help Jose sort the array using the merge sort algorithm.

Input Format

The first line of input consists of an integer N, representing the number of fractions to be sorted.

The second line consists of N double-point numbers, separated by spaces, representing the fractions array.

Output Format

The output prints N double-point numbers, sorted in increasing order, and rounded to three decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
0.123 0.543 0.321 0.789

Output: 0.123 0.321 0.543 0.789

Answer

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


int compare(double a, double b) {
   if (a < b) return -1;
   if (a > b) return 1;
   return 0;
}
void merge(double arr[], int l, int m, int r) {
   int n1 = m - l + 1;
   int n2 = r - m;

   double L[n1], R[n2];

   for (int i = 0; i < n1; i++)
      L[i] = arr[l + i];
   for (int j = 0; j < n2; j++)
      R[j] = arr[m + 1 + j];

   int i = 0, j = 0, k = l;

   while (i < n1 && j < n2) {
      if (compare(L[i], R[j]) <= 0)
         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(double arr[], int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}
int main() {
    int n;
    scanf("%d", &n);
    double fractions[n];
    for (int i = 0; i < n; i++) {
        scanf("%lf", &fractions[i]);
    }
    mergeSort(fractions, 0, n - 1);
    for (int i = 0; i < n; i++) {
        printf("%.3f ", fractions[i]);
    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 20

Section 1 : MCQ

1. Which of the following best describes linear probing in hashing?

Answer

Resolving collisions by linearly searching for the next free slot

Status : Correct ***Marks : 1/1***

2. In the folding method, what is the primary reason for reversing alternate parts before addition?

Answer

To reduce the chance of collisions caused by similar digit patterns

Status : Correct ***Marks : 1/1***

3. Which of the following statements is TRUE regarding the folding method?

Answer

It divides the key into parts and adds them.

Status : Correct ***Marks : 1/1***

4. Which of the following values of 'm' is recommended for the division method in hashing?

Answer

A prime number

Status : Correct ***Marks : 1/1***

5. In linear probing, if a collision occurs at index i, what is the next index checked?

Answer

(i + 1) % table_size

Status : Correct ***Marks : 1/1***

6. Which situation causes clustering in linear probing?

Answer

All the mentioned options

Status : Correct ***Marks : 1/1***

7. What is the primary disadvantage of linear probing?

Answer

Clustering

Status : Correct ***Marks : 1/1***

8. What happens if we do not use modular arithmetic in linear probing?

Answer

Index goes out of bounds

Status : Correct ***Marks : 1/1***

9. What is the output of the mid-square method for a key k = 123 if the hash table size is 10 and you extract the middle two digits of k * k?

Answer

1

Status : Correct ***Marks : 1/1***

10. Which data structure is primarily used in linear probing?

Answer

Array

Status : Correct ***Marks : 1/1***

11. Which folding method divides the key into equal parts, reverses some of them, and then adds all parts?

Answer

Folding reversal method

Status : Correct ***Marks : 1/1***

12. What does a deleted slot in linear probing typically contain?

Answer

A special "deleted" marker

Status : Correct ***Marks : 1/1***

13. What is the initial position for a key k in a linear probing hash table?

Answer

k % table_size

Status : Correct ***Marks : 1/1***

14. In division method, if key = 125 and m = 13, what is the hash index?

Answer

8

Status : Correct ***Marks : 1/1***

15. What is the worst-case time complexity for inserting an element in a hash table with linear probing?

Answer

O(n)

Status : Correct ***Marks : 1/1***

16. In the division method of hashing, the hash function is typically written as:

Answer

h(k) = k % m

Status : Correct ***Marks : 1/1***

17. Which of these hashing methods may result in more uniform distribution with small keys?

Answer

Mid-Square

Status : Correct ***Marks : 1/1***

18. What would be the result of folding 123456 into three parts and summing: (12 + 34 + 56)?

Answer

102

Status : Correct ***Marks : 1/1***

19. In C, how do you calculate the mid-square hash index for a key k, assuming we extract two middle digits and the table size is 100?

Answer

((k * k) / 100) % 100

Status : Correct ***Marks : 1/1***

20. Which C statement is correct for finding the next index in linear probing?

Answer

index = (index + 1) % size;

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ravi is building a basic hash table to manage student roll numbers for quick lookup. He decides to use Linear Probing to handle collisions.

Implement a hash table using linear probing where:

The hash function is: index = roll_number % table_sizeOn collision, check subsequent indexes (i+1, i+2, ...) until an empty slot is found.

You need to:

Insert a list of n student roll numbers into the hash table.Print the final state of the hash table. If a slot is empty, print -1.

Input Format

The first line of the input contains two integers n and table_size, where n is the

number of roll numbers to be inserted, and table_size is the size of the hash table.

The second line contains n space-separated integers — the roll numbers to insert into the hash table.

Output Format

The output should print a single line with table_size space-separated integers representing the final state of the hash table after all insertions.

If any slot remains unoccupied, it should be represented as -1.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4 7
50 700 76 85

Output: 700 50 85 -1 -1 -1 76

Answer

```
#include <stdio.h>

#define MAX 100

void initializeTable(int table[], int size) {
    for (int i = 0; i < size; i++) {
        table[i] = -1;
    }
}

int linearProbe(int table[], int size, int num) {
    int index = num % size;
    while (table[index] != -1) {
        index = (index + 1) % size;
    }
    return index;
}

void insertIntoHashTable(int table[], int size, int arr[], int n) {
```

```
    for (int i = 0; i < n; i++) {
        int index = linearProbe(table, size, arr[i]);
        table[index] = arr[i];
    }
}

void printTable(int table[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d", table[i]);
        if (i < size - 1) printf(" ");
    }
    printf("\n");
}

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

    int arr[MAX];
    int table[MAX];

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

    initializeTable(table, table_size);
    insertIntoHashTable(table, table_size, arr, n);
    printTable(table, table_size);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Priya is developing a simple student management system. She wants to store roll numbers in a hash table using Linear Probing, and later search for specific roll numbers to check if they exist.

Implement a hash table using linear probing with the following operations:

Insert all roll numbers into the hash table.For a list of query roll numbers, print "Value x: Found" or "Value x: Not Found" depending on whether it exists in the table.

Input Format

The first line contains two integers, n and table_size — the number of roll numbers to insert and the size of the hash table.

The second line contains n space-separated integers — the roll numbers to insert.

The third line contains an integer q — the number of queries.

The fourth line contains q space-separated integers — the roll numbers to search for.

Output Format

The output print q lines — for each query value x, print: "Value x: Found" or "Value x: Not Found"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5 10
21 31 41 51 61
3
31 60 51

Output: Value 31: Found
Value 60: Not Found
Value 51: Found

Answer

```
#include <stdio.h>

#define MAX 100

void initializeTable(int table[], int size) {
    for (int i = 0; i < size; i++) {
        table[i] = -1;
    }
}

int linearProbe(int table[], int size, int num) {
    int index = num % size;
    while (table[index] != -1 && table[index] != num) {
        index = (index + 1) % size;
    }
```

```
    return index;
}

void insertIntoHashTable(int table[], int size, int arr[], int n) {
    for (int i = 0; i < n; i++) {
        int index = linearProbe(table, size, arr[i]);
        table[index] = arr[i];
    }
}

int searchInHashTable(int table[], int size, int num) {
    int index = num % size;
    int start = index;
    while (table[index] != -1) {
        if (table[index] == num)
            return 1;
        index = (index + 1) % size;
        if (index == start)
            break;
    }
    return 0;
}

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

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

    initializeTable(table, table_size);
    insertIntoHashTable(table, table_size, arr, n);

    int q, x;
    scanf("%d", &q);
    for (int i = 0; i < q; i++) {
        scanf("%d", &x);
        if (searchInHashTable(table, table_size, x))
            printf("Value %d: Found\n", x);
        else
            printf("Value %d: Not Found\n", x);
    }
```

```
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a messaging application, users maintain a contact list with names and corresponding phone numbers. Develop a program to manage this contact list using a dictionary implemented with hashing.

The program allows users to add contacts, delete contacts, and check if a specific contact exists. Additionally, it provides an option to print the contact list in the order of insertion.

Input Format

The first line consists of an integer n, representing the number of contact pairs to be inserted.

Each of the next n lines consists of two strings separated by a space: the name of the contact (key) and the corresponding phone number (value).

The last line contains a string k, representing the contact to be checked or removed.

Output Format

If the given contact exists in the dictionary:

1. The first line prints "The given key is removed!" after removing it.
2. The next n - 1 lines print the updated contact list in the format: "Key: X; Value: Y" where X represents the contact's name and Y represents the phone number.

If the given contact does not exist in the dictionary:

1. The first line prints "The given key is not found!".
2. The next n lines print the original contact list in the format: "Key: X; Value: Y" where X represents the contact's name and Y represents the phone number.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 3
Alice 1234567890
Bob 9876543210
Charlie 4567890123
Bob

Output: The given key is removed!
Key: Alice; Value: 1234567890
Key: Charlie; Value: 4567890123

Answer

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

#define MAX 50

char names[MAX][11];
```

```
char numbers[MAX][20];
int count = 0;

int findIndex(char key[]) {
    for (int i = 0; i < count; i++) {
        if (strcmp(names[i], key) == 0)
            return i;
    }
    return -1;
}

void insertContact(char name[], char number[]) {
    strcpy(names[count], name);
    strcpy(numbers[count], number);
    count++;
}

void removeContact(char key[]) {
    int idx = findIndex(key);
    if (idx == -1) {
        printf("The given key is not found!");
        for (int i = 0; i < count; i++)
            printf("\nKey: %s; Value: %s", names[i], numbers[i]);
    } else {
        printf("The given key is removed!");
        for (int i = idx; i < count - 1; i++) {
            strcpy(names[i], names[i + 1]);
            strcpy(numbers[i], numbers[i + 1]);
        }
        count--;
        for (int i = 0; i < count; i++)
            printf("\nKey: %s; Value: %s", names[i], numbers[i]);
    }
}

int main()
{
    int n;
    scanf("%d", &n);
    char name[11], number[20];
    for (int i = 0; i < n; i++) {
        scanf("%s %s", name, number);
```

```
        insertContact(name, number);
    }
    char key[11];
    scanf("%s", key);
    removeContact(key);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Develop a program using hashing to manage a fruit contest where each fruit is assigned a unique name and a corresponding score. The program should allow the organizer to input the number of fruits and their names with scores.

Then, it should enable them to check if a specific fruit, identified by its name, is part of the contest. If the fruit is registered, the program should display its score; otherwise, it should indicate that it is not included in the contest.

Input Format

The first line consists of an integer N, representing the number of fruits in the contest.

The following N lines contain a string K and an integer V, separated by a space, representing the name and score of each fruit in the contest.

The last line consists of a string T, representing the name of the fruit to search for.

Output Format

If T exists in the dictionary, print "Key "T" exists in the dictionary.".

If T does not exist in the dictionary, print "Key "T" does not exist in the dictionary.".

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 2
banana 2
apple 1
Banana

Output: Key "Banana" does not exist in the dictionary.

Answer

```
#include<stdio.h>
#include<string.h>
struct fruit{
    char name[100];
    int num;

};
int main()
{
    int n;
    scanf("%d",&n);
    struct fruit fruits[20];
    for(int i=0;i<n;i++)
    {
      scanf("%s %d",fruits[i].name,&fruits[i].num) ;
    }
```

```
    char source[100];
    scanf("%s",source);
    int found=0;
    for(int i=0;i<n;i++){
        if(strcmp(fruits[i].name,source)==0){
            found=1;
            break;
        }
    }
    if(found){
        printf("Key \"%s\" exists in the dictionary.",source);
    }else{
        printf("Key \"%s\" does not exist in the dictionary.",source);
    }
}
```

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

Rajalakshmi Engineering College

Name: Ashvanth M
Email: 241501024@rajalakshmi.edu.in
Roll no: 241501024
Phone: 9080460667
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are provided with a collection of numbers, each represented by an array of integers. However, there's a unique scenario: within this array, one element occurs an odd number of times, while all other elements occur an even number of times. Your objective is to identify and return the element that occurs an odd number of times in this arrangement.

Utilize mid-square hashing by squaring elements and extracting middle digits for hash codes. Implement a hash table for efficient integer occurrence tracking.

Note: Hash function: squared = key * key.

Example

Input:

7

2 2 3 3 4 4 5

Output:

5

Explanation

The hash function and the calculated hash indices for each element are as follows:

2 -> hash(2*2) % 100 = 4

3 -> hash(3*3) % 100 = 9

4 -> hash(4*4) % 100 = 16

5 -> hash(5*5) % 100 = 25

The hash table records the occurrence of each element's hash index:

Index 4: 2 occurrences

Index 9: 2 occurrences

Index 16: 2 occurrences

Index 25: 1 occurrence

Among the elements, the integer 5 occurs an odd number of times (1 occurrence) and satisfies the condition of the problem. Therefore, the program outputs 5.

Input Format

The first line of input consists of an integer N, representing the size of the array.

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

Output Format

The output prints a single integer representing the element that occurs an odd

number of times.

If no such element exists, print -1.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 7
2 2 3 3 4 4 5

Output: 5

Answer

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

#define MAX_SIZE 100


unsigned int hash(int key, int tableSize) {
    int square = key * key;

    // Convert square to string
    char squareStr[20];
    sprintf(squareStr, "%d", square);
    int len = strlen(squareStr);

    int mid;
    if (len % 2 == 0) {
        int midStart = len / 2 - 1;
        char midDigits[3];
        midDigits[0] = squareStr[midStart];
        midDigits[1] = squareStr[midStart + 1];
        midDigits[2] = '\0';
        mid = atoi(midDigits);
    } else {
        int midStart = len / 2;
```

```
        mid = squareStr[midStart] - '0';
    }

    return mid % tableSize;
}

int getOddOccurrence(int arr[], int size) {
    int hashTable[MAX_SIZE] = {0};

    // Count occurrences via hash
    for (int i = 0; i < size; i++) {
        unsigned int idx = hash(arr[i], MAX_SIZE);
        hashTable[idx]++;
    }

    // Find element with odd occurrence using its hash
    for (int i = 0; i < size; i++) {
        unsigned int idx = hash(arr[i], MAX_SIZE);
        if (hashTable[idx] % 2 != 0) {
            return arr[i];
        }
    }

    return -1;
}

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

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

    printf("%d\n", getOddOccurrence(arr, n));

    return 0;
}
```

Status : Correct *Marks : 10/10*