

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 0_Arrays and Functions

Attempt : 1
Total Mark : 5
Marks Obtained : 4

Section 1 : Coding

1. Problem Statement

Saurabh is the manager of a growing tech company. He needs a program to record and analyze the monthly salaries of his employees. The program will take the number of employees and their respective salaries as input and then calculate the average salary, and find the highest and lowest salary among them.

Help Saurabh automate this task efficiently.

Input Format

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

The second line consists of n integers, where each integer represents the salary of an employee.

Output Format

The output prints n lines, where each line will display: "Employee i: "Salary

Where i is the employee number (starting from 1) and salary is the respective salary of that employee.

After that, print the average salary in the following format: "Average Salary: "average_salary

Where average_salary is the average salary of all employees, rounded to two decimal places.

Next, print the highest salary in the following format: "Highest Salary: "max_salary

Where max_salary is the highest salary among all employees.

Finally, print the lowest salary in the following format: "Lowest Salary: "min_salary

Where min_salary is the lowest salary among all employees.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

4000

3500

6000

2500

4500

Output: Employee 1: 4000

Employee 2: 3500

Employee 3: 6000

Employee 4: 2500

Employee 5: 4500

Average Salary: 4100.00

Highest Salary: 6000

Lowest Salary: 2500

Answer

```
#include<stdio.h>
int main(){
    int a;
    scanf("%d",&a);
    int arr[a];
    for(int i=0;i<a;i++){
        scanf("%d",&arr[i]);
    }
    float tot=0;
    for (int j=0;j<a;j++){
        printf("Employee %d:%d\n",j+1,arr[j]);
        tot+=arr[j];
    }
    float avg=tot/a;
    for(int i=0;i<a;i++){
        for(int j=i+1;j<a;j++){
            if(arr[i]>arr[j]){
                int min=arr[j];
                arr[j]=arr[i];
                arr[i]=min;
            }
        }
    }

    printf("Average Salary:%.2f\n",avg);

    printf("Highest Salary:%d\n",arr[a-1]);
    printf("Lowest Salary:%d\n",arr[0]);
```

```
return 0;
```

```
}
```

Status : Correct

Marks : 1/1

2. Problem Statement

Alex, a budding programmer, is tasked with writing a menu-driven program to perform operations on an array of integers. The operations include finding the smallest number, the largest number, the sum of all numbers, and their average. The program must repeatedly display the menu until Alex chooses to exit.

Write a program to ensure the specified tasks are implemented based on Alex's choices.

Input Format

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

The second line contains n space-separated integers representing the array elements.

The subsequent lines contain integers representing the menu choices:

Choice 1: Find and display the smallest number.

Choice 2: Find and display the largest number.

Choice 3: Calculate and display the sum of all numbers.

Choice 4: Calculate and display the average of all numbers as double.

Choice 5: Exit the program.

Output Format

For each valid menu choice, print the corresponding result:

For choice 1, print "The smallest number is: X", where X is the smallest number in the array.

For choice 2, print "The largest number is: X", where X is the largest number in the array.

For choice 3, print "The sum of the numbers is: X", where X is the sum of all numbers in the array.

For choice 4, print "The average of the numbers is: X. XX", where X.XX is the double value representing an average of all numbers in the array, rounded to two decimal places.

For choice 5, print "Exiting the program".

If an invalid choice is made, print "Invalid choice! Please enter a valid option (1-5)."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3

10 20 30

1

5

Output: The smallest number is: 10

Exiting the program

Answer

-

Status : Skipped

Marks : 0/1

3. Problem Statement

Tim is creating a program to track and analyze student attendance. The program requires two inputs: the total number of students (n) and the total number of class sessions (m). The task is to design and populate an

attendance matrix, 'matrix', representing the attendance record of each student for each session.

The program's specific objective is to determine whether the last student on the list attended an even or odd number of classes. This functionality will aid teachers in quickly evaluating the attendance habits of individual students.

Input Format

The first line of input consists of a positive integer n, representing the number of students.

The second line consists of a positive integer m, representing the number of class sessions.

The next n lines consist of m space-separated positive integers representing the number of classes attended by the student.

Output Format

The output displays one of the following results:

If the last session is even the output prints "[LastSession] is even".

If the last session is odd the output prints "[LastSession] is odd".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 2

2

1 2

3 100

Output: 100 is even

Answer

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

```
int main(){
```

```
    int n,m,a;
```

```

scanf("%d\n%d",&n,&m);
int matrix[n][m];
for (int i=0;i<n;i++){
    for(int j=0;j<m;j++){
        scanf("%d",&matrix[i][j]);
    }
}
a=matrix[n-1][m-1];
if(a%2==0){
    printf("%d is even",a);
}
else
    printf("%d is odd", a);
}

```

Status : Correct

Marks : 1/1

4. Problem Statement

Write a program that will read a Matrix (two-dimensional arrays) and print the sum of all elements of each row by passing the matrix to a function.

Function Signature: void calculateRowSum(int [][], int, int)

Input Format

The first line consists of an integer M representing the number of rows.

The second line consists of an integer N representing the number of columns.

The next M lines consist of N space-separated integers in each line representing the elements of the matrix.

Output Format

The output displays the sum of all elements of each row separated by a space.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3

3

1 2 3

4 5 6

7 8 9

Output: 6 15 24

Answer

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

```
void calculateRowSum (int matrix[20][20], int rows, int cols){
```

```
    for(int i=0;i<rows;i++){
```

```
        int a=0;
```

```
        for(int j=0;j<cols;j++){
```

```
            a+=matrix[i][j];
```

```
        }
```

```
        printf("%d",a);
```

```
        printf("\t");
```

```
    }
```

```
}
```

```
int main() {
```

```
    int matrix[20][20];
```

```
    int r, c;
```

```
    scanf("%d", &r);
```

```
    scanf("%d", &c);
```

```
    for (int i = 0; i < r; i++) {
```

```
        for (int j = 0; j < c; j++) {
```

```
            scanf("%d", &matrix[i][j]);
```

```
        }
```

```
    }
```

```
    calculateRowSum(matrix, r, c);
```

```
    return 0;
```

```
}
```

Status : Correct

Marks : 1/1

5. Problem Statement

Write a program that reads an integer 'n' and a square matrix of size 'n x n' from the user. The program should then set all the elements in the lower triangular part of the matrix (including the main diagonal) to zero using a function and display the resulting matrix.

Function Signature: void setZeros(int [][], int)

Input Format

The first line consists of an integer M representing the number of rows & columns.

The next M lines consist of M space-separated integers in each line representing the elements of the matrix.

Output Format

The output displays the matrix containing M space-separated elements in M lines where the lower triangular elements are replaced with zero.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
10 20 30
40 50 60
70 80 90
Output: 0 20 30
0 0 60
0 0 0

Answer

```
#include <stdio.h>

void setZeros(int arr[10][10], int n){
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
```

```

        if(j<=i){
            arr[i][j]=0;
        }
    }
}

int main() {
    int arr1[10][10];
    int n;

    scanf("%d", &n);

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

    setZeros(arr1, n);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", arr1[i][j]);
        }
        printf("\n");
    }

    return 0;
}

```

Status : Correct

Marks : 1/1

Rajalakshmi Engineering College

Name: R.R CHANDRU

Email: 240701087@rajalakshmi.edu.in

Roll no: 240701087

Phone: 9003697366

Branch: REC

Department: I CSE AG

Batch: 2028

Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 0_Pointers

Attempt : 1

Total Mark : 5

Marks Obtained : 5

Section 1 : Coding

1. Problem Statement

Ria is a mathematician who loves exploring combinatorics. She is working on a project that involves calculating permutations.

Ria wants to create a program that takes the values of n and r as input and calculates the permutations of n elements taken r at a time.

Write a program using pointers and a function calculatePermutations that, given the values of n and r , calculates and prints the permutations of n elements taken r at a time.

Permutation: $n! / (n - r)!$

Input Format

The first line consists of an integer n, representing the total number of elements.

The second line consists of an integer r, representing the number of elements to be taken at a time.

Output Format

The output prints the result of the permutation.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4

3

Output: 24

Answer

```
#include<stdio.h>
long fact(int n,int r)
{
    long f=1;
    for(int i=0;i<r;i++)
        f*=(n-i);
    return f;
}
int main(){
    int n,r;
    scanf("%d",&n);
    scanf("%d",&r);
    long X=fact(n,r);
    printf("%ld\n",X);
    return 0;
}
```

Status : Correct

Marks : 1/1

2. Problem Statement

Raj wants to create a program using pointers and a structure named Employee to manage employee information.

He seeks your assistance to input the employee's name, salary, and hours worked. Implement a salary increase based on hours worked, and calculate the final salary. Calculate the total salary for 30 days. Display the results of the final and total salary.

Salary increase criteria:

If hours worked ≥ 12 , the increase is Rs. 150.00. If hours worked ≥ 10 , but less than 12, the increase is Rs. 100.00. If hours worked ≥ 8 , but less than 10, the increase is Rs. 50.00. If hours worked < 8 , there is no increase.

Input Format

The first line of input consists of a string, representing the Employee's name.

The second line consists of a double-point number, representing the Employee's current salary.

The third line consists of an integer, representing the number of hours worked by the employee.

Output Format

The first line of output prints "Final Salary: Rs. " followed by a double value, representing the final salary, rounded off to two decimal places.

The second line prints "Total Salary: Rs. " followed by a double value, representing the total salary for 30 days, rounded off to two decimal places.

Refer to the sample outputs for formatting specifications.

Sample Test Case

Input: Akil

3000.00

6

Output: Final Salary: Rs. 3000.00

Total Salary: Rs. 90000.00

Answer

```
#include<stdio.h>
#include<string.h>
struct Employee
{
    char name[51];
    double sal;
    int hrs;
};
double cfs(double bs, int hrs){
    double inc=0.0;
    if(hrs >=12)
        inc=150.00;
    else if(hrs>=10)
        inc=100.00;
    else if(hrs >=8)
        inc =50.00;
    return bs+inc;
}
int main(){
    struct Employee emp;
    scanf("%50s %lf %d",emp.name,&emp.sal,&emp.hrs);
    double fs=cfs(emp.sal,emp.hrs);
    double ts=fs*30;
    printf("Final Salary: Rs. %.2f\nTotal Salary:Rs. %.2f\n",fs,ts);
    return 0;
}
```

Status : Correct

Marks : 1/1

3. Problem Statement

Rajwinder wants a program to determine retirement details for a person based on their age.

Create a program that uses a structure called Person to hold the age as an attribute with a pointer.

If the age is under 18, display "Invalid".If the age is 65 or older, print "Already retired!".Otherwise, calculate and output the retirement year,

remaining years, and remaining days until retirement.

Note: Age 65 is considered as retirement age. Assume the current year as 2023 and there are 365 days per year for calculation.

Input Format

The input consists of an integer representing the person's age.

Output Format

If the age is under 18, the output displays "Invalid" and terminates.

If the age is 65 or older, the output displays "Already retired!" and terminates.

Otherwise, the output displays the following.

1. The first line displays "Retirement Year: " followed by an integer representing the retirement year.
2. The second line displays "Remaining Years: " followed by an integer representing the remaining years left for retirement.
3. The third line displays "Remaining Days: " followed by an integer representing the remaining days left for retirement.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 43

Output: Retirement Year: 2045

Remaining Years: 22

Remaining Days: 8030

Answer

```
#include<stdio.h>
typedef struct{
    int age;
}Person;
void det(Person*P){
    int cy=2023;
    int rage=65;
```

```

    if (P->age<18){
        printf("Invalid");
        return;
    }
    if(P->age>=rage){
        printf("Already Retired!");
        return;
    }
    int ryears=rage-P->age;
    int rdays=ryears*365;
    int reyear=cy+ryears;
    printf("Retirement Year: %d\n",reyear);
    printf("Remaining Years: %d\n",ryears);
    printf("Remaining Days: %d\n",rdays);
}
int main(){
    Person P;
    scanf("%d",&P.age);
    det(&P);
    return 0;
}

```

Status : Correct

Marks : 1/1

4. Problem Statement

Sam is developing a program for analyzing daily temperature fluctuations. Users input the number of days, followed by daily temperature values whose memory is allocated using malloc.

The program calculates and displays the following:

The absolute temperature changes between consecutive days (The first value remains the same). The average temperature of adjacent days.

This allows users to gain insights into daily temperature variations for better analysis.

For Example,

Let us assume the temperature for 3 days as 25.5, 28.0, and 23.5.

The absolute differences:

Day 1: (N/A, as there is no previous day) = 25.50
Day 2: $\text{abs}(28.0 - 25.5) = 2.50$
Day 3: $\text{abs}(23.5 - 28.0) = 4.50$

The average temperatures:

Day 1: (N/A, as there is no previous day) = 25.50
Day 2: $(25.5 + 23.5) / 2.0 = 24.50$
Day 3: (N/A, as there is no next day) = 23.50

Input Format

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

The second line consists of N space-separated float values, representing the temperature values for N days.

Output Format

The first line displays the absolute temperature change for N days as float values, rounded to two decimal places, separated by a space.

The second line displays the average temperature for N days as float values, rounded to two decimal places, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
25.5 28.0 23.5
Output: 25.50 2.50 4.50
25.50 24.50 23.50

Answer

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main(){
    int n;
    scanf("%d",&n);
```

```

float *temp=(float*) malloc(n*sizeof(float));
for(int i=0;i<n;i++)
scanf("%f",&temp[i]);
printf("%.2f ",temp[0]);
for(int i=1;i<n;i++)
printf("%.2f ",fabs(temp[i]-temp[i-1]));
printf("\n");
printf("%.2f ",temp[0]);
for(int i=1;i<n-1;i++){
    printf("%.2f ",(temp[i-1]+temp[i+1])/2.0);
    temp[i]=(temp[i-1]+temp[i+1])/2.0;
}
printf("%.2f",temp[n-1]);
free(temp);
return 0;
}

```

Status : Correct

Marks : 1/1

5. Problem Statement

Daniel is working on a project that involves analyzing data stored in float arrays. He needs to determine whether a given float array contains only positive numbers.

To achieve this, he needs a program that can accurately evaluate the contents of float arrays using malloc().

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 float values, representing the elements of the array.

Output Format

If all the array elements are positive, print "All elements are positive."

If the array contains at least one positive element, print "At least one element is positive."

If there are no positive elements in the array, print "No positive elements in the array."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

50.0 -2.3 3.7 -4.8 5.2

Output: At least one element is positive.

Answer

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    int n;
    scanf("%d",&n);
    float *arr=(float*)malloc(n*sizeof(float));
    for(int i=0;i<n;i++)
        scanf("%f",&arr[i]);
    int ap=1, hp=0;
    for(int i=0;i<n;i++){
        if(arr[i]>0){
            hp=1;
        }
        else{
            ap=0;
        }
    }
    if(ap)
        printf("All elements are positive.");
    else if(hp)
        printf("At Least one element is positive.");
    else
        printf("No positive elements in the array.");
    free(arr);
    return 0;
}
```

Status : Correct

Marks : 1/1

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_MCQ

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : MCQ

1. 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

2. 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

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

Answer

Binary search

Status : Correct

Marks : 1/1

4. 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

5. 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

6. 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

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. 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

9. 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

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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 1

Attempt : 3
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

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

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

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

Output Format

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

Sample Test Case

Input: 3

2 2

3 1

4 0

3

2 2

3 1

4 0

Output: 18

Answer

```
// You are using GCC
```

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

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

```
typedef struct Node
```

```
{
```

```
    int coeff,exp;
```

```
    struct Node* next;
```

```
}Node;
```

```
Node* createNode(int coeff,int exp)
```

```
{
```

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

```
    newNode->coeff=coeff;
```

```
    newNode->exp=exp;
```

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

```
    return newNode;
```

```
}
```

```
void insertTerm(Node** poly,int coeff,int exp)
```

```
{
```

```
    Node* newNode=createNode(coeff,exp);
```

```
    if(!*poly)
```

```

    {
        *poly=newNode;
    }
    else
    {
        Node*temp=*poly;
        while(temp->next) temp = temp->next;
        temp->next=newNode;
    }
}

int sumOfCoefficients(Node* poly)
{
    int sum=0;
    while (poly)
    {
        sum+=poly->coeff;
        poly=poly->next;
    }
    return sum;
}

Node* addPolynomials(Node* poly1, Node* poly2)
{
    Node* result=NULL;
    while(poly1 || poly2)
    {
        if(poly1 &&(!poly2 || poly1->exp >poly2->exp))
        {
            insertTerm(&result,poly1->coeff,poly1->exp);
            poly1=poly1->next;
        }
        else if(poly2 &&(!poly1 || poly1->exp >poly2->exp))
        {
            insertTerm(&result,poly2->coeff,poly2->exp);
            poly2=poly2->next;
        }
        else
        {
            insertTerm(&result,poly1->coeff+poly2->coeff,poly1->exp);
            poly1=poly1->next;
            poly2=poly2->next;
        }
    }
}

```

```

    }
    return result;
}
int main()
{
    int n,m;
    Node* poly1=NULL, *poly2=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        int coeff,exp;
        scanf("%d %d",&coeff,&exp);
        insertTerm(&poly1,coeff,exp);
    }
    scanf("%d",&m);
    for(int i=0;i<m;i++)
    {
        int coeff,exp;
        scanf("%d %d",&coeff,&exp);
        insertTerm(&poly2,coeff,exp);
    }
    Node* result = addPolynomials(poly1,poly2);
    printf("%d\n",sumOfCoefficients(result));
    return 0;
}

```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 4

Attempt : 2
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

The second line consists of N space-separated integers.

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

78 89 34 51 67

Output: 67 51 34 89 78

Answer

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

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

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

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

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

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

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

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

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

    return 0;
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU

Email: 240701087@rajalakshmi.edu.in

Roll no: 240701087

Phone: 9003697366

Branch: REC

Department: I CSE AG

Batch: 2028

Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 5

Attempt : 2

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>
```

```
struct node
```

```
{
```

```
    float data;
```

```
    struct node *next;
```

```
} *head=NULL;
```

```
typedef struct node node;
```

```
void insert(float value)
```

```
{
```

```
    node* newnode=(node*)malloc(sizeof(node));
```

```
    newnode->data=value;
```

```
    newnode->next=head;
```

```
    head=newnode;
```

```
}
```

```
void display()
```

```
{
```

```
    node*temp=head;
```

```

while (temp!=NULL)
{
    printf("GPA: %.1f\n",temp->data);
    temp=temp->next;
}
printf("\n");
}
int length()
{
    int count=0;
    node*temp=head;
    while(temp!=NULL)
    {
        count++;
        temp=temp->next;
    }
    return count;
}
void delbeg()
{
    if(head!=NULL)
    {
        node*tempnode=head;
        head=head->next;
        free(tempnode);
    }
}
void delend()
{
    if(head==NULL)
        return;
    if(head->next==NULL)
    {
        free(head);
        head=NULL;
        return ;
    }
    node*temp=head;
    while(temp->next->next!=NULL)
        temp=temp->next;
    free(temp->next);
    temp->next=NULL;
}

```

```

}
void delmid(int position)
{
    if(position==1)
    {
        delbeg();
        return;
    }
    node* temp=head;
    node* prev=NULL;
    int count=1;
    while(temp!=NULL && count<position)
    {
        prev=temp;
        temp=temp->next;
        count++;
    }
    if(temp==NULL)
        return;
    prev->next=temp->next;
    free(temp);
}
void del(int pos)
{
    int n=length();
    if(pos<1 || pos>n)
    {
        printf("Invalid position\n");
        return;
    }
    if(pos==1)
    {
        delbeg();
    }
    else if(pos==n)
    {
        delend();
    }
    else
    {
        delmid(pos);
    }
}

```

```
    display();
}
int main()
{
    int n,a;
    float values;
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        scanf("%f",&values);
        insert(values);
    }
    scanf("%d",&a);
    del(a);
    return 0;
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 6

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

23 85 47 62 31

Output: 23 85 47 62 31

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct Node
{
    int rollnumber;
    struct Node*next;
};
typedef struct Node node;
void display(node* head)
{
    node* temp=head;
    while(temp!=NULL)
    {
        printf("%d",temp->rollnumber);
        temp=temp->next;
    }
    printf("\n");
}
node* insertAtEnd(node* head, int roll)
{
    node *newnode=(node*)malloc(sizeof(node));
    newnode->rollnumber=roll;
    newnode->next=NULL;
    if(head==NULL)
    {
        return newnode;
```

```
}
node* position=head;
while(position->next!=NULL)
{
    position=position->next;
}
position->next=newnode;
return head;
}
int main()
{
    node* head=NULL;
    int n,roll;
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        scanf("%d",&roll);
        head=insertAtEnd(head,roll);
    }
    display(head);
    return 0;
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 7

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

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

Input Format

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

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

Output Format

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

10 20 30 40 50

Output: 50 40 30 20 10

Middle Element: 30

Answer

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

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

```
struct Node {
```

```
    int data;
```

```
    struct Node* next;
```

```
};
```

```
struct Node* push(struct Node* head,int value) {
```

```
    struct Node* newnode = (struct Node*) malloc(sizeof(struct Node));
```

```
    newnode->next = head;
```

```
    newnode->data = value;
```

```
    return newnode;
```

```
}
```

```
int printMiddle(struct Node* head) {
```

```
    int len = 0;
```

```
    struct Node* temp = head;
```

```
    while(temp != NULL) {
```

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

```
int main() {  
    struct Node* head = NULL;  
    int n;  
    scanf("%d", &n);  
    int value;  
  
    for (int i = 0; i < n; i++) {  
        scanf("%d", &value);  
        head = push(head, value);  
    }  
  
    struct Node* current = head;  
    while (current != NULL) {  
        printf("%d ", current->data);  
        current = current->next;  
    }  
    printf("\n");  
}
```

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

```
current = head;  
while (current != NULL) {  
    struct Node* temp = current;  
    current = current->next;  
    free(temp);  
}  
  
return 0;
```

}

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 17

Section 1 : MCQ

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

Answer

2

Status : Correct

Marks : 1/1

2. 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

3. How do you reverse a doubly linked list?

Answer

By swapping the next and previous pointers of each node

Status : Correct

Marks : 1/1

4. 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

5. 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

6. 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 first occurrence of a given data value in a doubly linked list.

Status : Correct

Marks : 1/1

7. 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

8. 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

9. 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

10. 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

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

Status : Wrong

Marks : 0/1

11. 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

12. 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

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

Answer

prev

Status : Correct

Marks : 1/1

14. 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

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

Answer

Two-way linked lists are faster for insertion and deletion operations.

Status : Wrong

Marks : 0/1

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

Answer

It will break the list

Status : Wrong

Marks : 0/1

17. 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

18. 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

19. 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

20. 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

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

Here are the main functionalities of the program:

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

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

Input Format

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

The input is terminated by entering - (hyphen).

Output Format

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: a b c -

Output: Forward Playlist: a b c

Backward Playlist: c b a

Answer

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

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

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

```
void insertAtEnd(struct Node** head, char item) {  
    Node * temp=(Node *)malloc(sizeof(Node));  
    temp->prev=NULL;  
    temp->next=NULL;  
    temp->item=item;  
    if(*head==NULL)  
    {  
        *head=temp;  
    }  
    else
```

```

{
    Node * p=*head;
    while(p->next!=NULL)
    {
        p=p->next;
    }
    p->next=temp;
    temp->prev=p;
}
//type your code here
}

void displayForward(struct Node* head) {
    //type your code here
    Node * p=head;
    while(p!=NULL)
    {
        printf("%c ",p->item);
        p=p->next;
    }
    printf("\n");
}

void displayBackward(struct Node* tail) {
    //type your code here
    Node * p=tail;
    while(p!=NULL)
    {
        printf("%c ",p->item);
        p=p->prev;
    }
    printf("\n");
}

void freePlaylist(struct Node* head) {
    //type your code here
    Node * p=head;
    while(p!=NULL)
    {
        Node *a=p->next;
        free(p);
        p=a;
    }
}

```

```
}
```

```
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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

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

Output Format

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

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3

163 137 155

Output: 163

Answer

```
// You are using GCC
```

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

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

```
struct node
```

```
{
```

```
    int data;
```

```
    struct node * prev;
```

```
    struct node * next;
```

```
};
```

```
typedef struct node node;
```

```
void insert(node **head,int data)
```

```
{
```

```
    node * temp=(node *)malloc(sizeof(node));
```

```
    temp->prev=NULL;
```

```
    temp->next=NULL;
```

```
    temp->data=data;
```

```
    if(*head==NULL)
```

```
    {
```

```
        *head=temp;
```

```
    }
```

```
    else
    {
        node * p=*head;
        while(p->next!=NULL)
        {
            p=p->next;
        }
        p->next=temp;
        temp->prev=p;
    }
}
```

```
void display(node *head)
{
    if(head==NULL)
    {
        printf("Empty list!");
        return;
    }
    node * a=head;
    int b=a->data;
    a=a->next;
    while(a!=NULL)
    {
        if(a->data>b)
        {
            b=a->data;
        }
        a=a->next;
    }
    printf("%d",b);
}
```

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

    node * head=NULL;
    int b;
    for(int i=0;i<n;i++)
```

```
{
    scanf("%d",&b);
    insert(&head,b);
}
display(head);
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 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 GCC
```

```
void traverse() {
    //type your code here
    printf("Node Inserted\n");
    struct node * a=start;
    while(a!=NULL)
    {
        printf("%d ",a->info);
        a=a->next;
```

```

    }
    printf("\n");
}

void insertAtFront(int data) {
    //type your code here
    struct node * temp=(node *)malloc(sizeof(node));
    temp->info=data;
    temp->next=NULL;
    temp->prev=NULL;

    if(start==NULL)
    {
        start=temp;
    }
    else
    {
        temp->next=start;
        temp->next->prev=temp;
    }
    start=temp;
}

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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

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

Input Format

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

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

10 20 30 40 50

Output: 10 20 30 40 50

Answer

```
// You are using GCC
```

```
// You are using GCC
```

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

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

```
struct node
```

```
{
```

```
    int data;
```

```
    struct node * next;
```

```
    struct node * prev;
```

```
};
```

```
typedef struct node node;
```

```
node * head=NULL;
```

```
void insert(int data)
```

```
{
```

```
    node * temp=(node *)malloc(sizeof(node));
```

```
    temp->data=data;
```

```
    temp->next=NULL;
```

```
    temp->prev=NULL;
```

```
    if(head==NULL)
```

```
    {
```



```
        head=temp;
    }
    else
    {
        node * a=head;
        while(a->next!=NULL)
        {
            a=a->next;
        }
        a->next=temp;
        temp->prev=a;
    }
}
```

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

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

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

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 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>
```

```
// Structure of a doubly linked list node
```

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

```
// Function to create a new node
```

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

```
// Function to insert an item at the end of the list
```

```
void insertAtEnd(struct Node** head, struct Node** tail, int data) {  
    struct Node* newNode = createNode(data);  
    if (*head == NULL) {  
        *head = *tail = newNode;  
        return;  
    }  
    (*tail)->next = newNode;  
    newNode->prev = *tail;  
    *tail = newNode;  
}
```

```
// Function to delete a node at a specific position
```

```
int deleteAtPosition(struct Node** head, struct Node** tail, int position) {  
    if (*head == NULL) return 0;
```

```
    struct Node* temp = *head;  
    int count = 1;
```

```
    // Traverse to the specified position
```

```
    while (temp != NULL && count < position) {  
        temp = temp->next;  
        count++;  
    }
```

```
// If position is invalid
if (temp == NULL) return 0;

// Update pointers to remove the node
if (temp->prev != NULL) temp->prev->next = temp->next;
if (temp->next != NULL) temp->next->prev = temp->prev;

// Update head or tail if necessary
if (temp == *head) *head = temp->next;
if (temp == *tail) *tail = temp->prev;

free(temp);
return 1;
}
```

```
// Function to display the list
void displayList(struct Node* head) {
    struct Node* temp = head;
    int count = 1;
    while (temp != NULL) {
        printf(" node %d : %d\n", count, temp->data);
        temp = temp->next;
        count++;
    }
}
```

```
int main() {
    struct Node *head = NULL, *tail = NULL;
    int n, data, p;
```

```
    // Input the number of items
    scanf("%d", &n);
```

```
    // Input the items and create the list
    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        insertAtEnd(&head, &tail, data);
    }
```

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

```
// Input the position to delete
scanf("%d", &p);

// Try to delete the item at the specified position
if (!deleteAtPosition(&head, &tail, p)) {
    printf("Invalid position. Try again.\n");
} else {
    // Display the list after deletion
    printf("\nAfter deletion the new list:\n");
    displayList(head);
}

return 0;
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 19

Section 1 : MCQ

1. 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

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. 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

4. 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

5. 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

6. 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

7. 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

8. 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

9. Elements are Added on _____ of the Stack.

Answer

Top

Status : Correct

Marks : 1/1

10. 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

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

Answer

142

Status : Correct

Marks : 1/1

12. 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

13. 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

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

Answer

All of the mentioned options

Status : Correct

Marks : 1/1

15. 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

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

Answer

Efficient memory usage

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

Answer

-18

Status : Correct

Marks : 1/1

19. 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 empty
Stack elements: 30 20 10
Stack Overflow
Stack elements: 50 40 30 20 10

Status : Correct

Marks : 1/1

20. 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

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

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

the stack.

Choice 2: Pop the integer from the stack.

Choice 3: Display the elements in the stack.

Choice 4: Exit the program.

Output Format

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

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

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

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

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

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

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

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

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

Sample Test Case

Input: 1 3

1 4

3

2

3

4

Output: Pushed element: 3

Pushed element: 4

Stack elements (top to bottom): 4 3

Popped element: 4

Stack elements (top to bottom): 3

Exiting program

Answer

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

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

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

```
struct Node* top = NULL;
```

```
void push(int data) {  
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  
    if (newNode == NULL) {  
        printf("Memory allocation failed\n");  
        return;  
    }  
    newNode->data = data;  
    newNode->next = top;  
    top = newNode;
```

```

    printf("Pushed element: %d\n", data);
}
void pop() {
    if (top == NULL) {
        printf("Stack is empty. Cannot pop.\n");
    } else {
        struct Node* temp = top;
        top = top->next;
        printf("Popped element: %d\n", temp->data);
        free(temp);
    }
}
void displayStack() {
    if (top == NULL) {
        printf("Stack is empty\n");
    } else {
        struct Node* current = top;
        printf("Stack elements (top to bottom): ");
        while (current != NULL) {
            printf("%d ", current->data);
            current = current->next;
        }
        printf("\n");
    }
}
int main() {
    int choice, value;
    do {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf("%d", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                displayStack();
                break;
            case 4:
                printf("Exiting program\n");

```

```
        return 0;
    default:
        printf("Invalid choice\n");
    }
} while (choice != 4);

return 0;
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

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

Input Format

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

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

Choice 2: Pop the book ID from the stack.

Choice 3: Display the book ID in the stack.

Choice 4: Exit the program.

Output Format

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

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

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

Sample Test Case

Input: 1 19

1 28

2

3

2

4

Output: Book ID 19 is pushed onto the stack

Book ID 28 is pushed onto the stack

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

Answer

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

```
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) {
        printf("Stack Overflow\n");
        return;
    }
    newNode->bookID = bookID;
    newNode->next = top;
    top = newNode;
    printf("Book ID %d is pushed onto the stack\n", bookID);
}
```

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

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

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

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

240701087

```
}  
}  
    return 0;  
}
```

240701087

240701087

240701087

Status : Correct

Marks : 10/10

240701087

240701087

240701087

240701087

240701087

240701087

240701087

240701087

240701087

240701087

240701087

240701087

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

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

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

Input Format

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

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

Choice 2: Pop the character from the stack.

Choice 3: Display the characters in the stack.

Choice 4: Exit the program.

Output Format

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2

4

Output: Stack is empty. Nothing to pop.

Answer

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

```
#include <stdbool.h>
```

```
#define MAX_SIZE 100
```

```
char items[MAX_SIZE];
```

```
int top = -1;
```

```
void initialize() {
```

```
    top = -1;
```

```
}
```

```
bool isFull() {
```

```
    return top == MAX_SIZE - 1;
```

```
}
```

```
bool isEmpty() {
```

```
    return top == -1;
```

```
}
```

```
void push(char value) {
```

```
    if (isFull()) {
```

```
        return;
```

```
    }
```

```
    items[++top] = value;
```

```
    printf("Pushed: %c\n", value);
```

```
}
```

```
char pop() {
```

```
    if (isEmpty()) {
```

```
        printf("Stack is empty. Nothing to pop.\n");
```

```
        return '\0';
```

```
    }
```

```
    printf("Popped: %c\n", items[top]);
```

```
    return items[top--];
```

```
}
```

```
void display() {
```

```
    if (isEmpty()) {
```

```
        printf("Stack is empty.\n");
```

```
        return;
```

```
    }
```

```
    printf("Stack elements: ");
```

```
    for (int i = top; i >= 0; i--) {
```

```
        printf("%c ", items[i]);
```

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

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

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

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Example

Input:

a+b

Output:

ab+

Explanation:

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

Input Format

The input is a string, representing the infix expression.

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: a+(b*e)

Output: abe*+

Answer

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

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

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

```

    return NULL;

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

    return stack;
}

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

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

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

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

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

int Prec(char ch) {
    if (ch == '+' || ch == '-') return 1;
    if (ch == '*' || ch == '/') return 2;
    if (ch == '^') return 3;
    return 0;
}

void infixToPostfix(char* exp) {
    struct Stack* stack = createStack(strlen(exp));
    if (!stack) {
        printf("Memory allocation failed\n");
        return;
    }

```

```

    }

    char output[100];
    int k = 0;

    for (int i = 0; exp[i]; i++) {
        char current = exp[i];

        if (isOperand(current)) {
            output[k++] = current;
        }

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

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

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

```

```

    while (!isEmpty(stack)) {
        output[k++] = pop(stack);
    }

```

```

    output[k] = '\0';
    printf("%s\n", output);

```

```

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

```



```
int main() {  
    char exp[100];  
    scanf("%s", exp);  
  
    infixToPostfix(exp);  
    return 0;  
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

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

Choice 2: Pop class schedule from the stack

Choice 3: Display the class schedules in the stack.

Choice 4: Exit the program.

Output Format

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

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

Refer to the sample output for the exact format.

Sample Test Case

Input: 1 d

1 h

3

2

3

4

Output: Adding Section: d

Adding Section: h

Enrolled Sections: h d

Removing Section: h

Enrolled Sections: d

Exiting program

Answer

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

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

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

```
struct Node* top = NULL;
```

```
void push(char value) {  
    if ((value < 'A' || value > 'Z') && (value < 'a' || value > 'z')) {  
        printf("Invalid input. Only alphabetic characters are allowed.\n");  
        return;  
    }  
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  
    if (!newNode) {  
        printf("Memory allocation failed.\n");  
        return;  
    }  
    newNode->data = value;  
    newNode->next = top;  
    top = newNode;  
    printf("Adding Section: %c\n", value);  
}
```

```
void pop() {  
    if (top == NULL) {  
        printf("Stack is empty. Cannot pop.\n");  
        return;  
    }  
    struct Node* temp = top;
```

```
    printf("Removing Section: %c\n", top->data);
    top = top->next;
    free(temp);
}
```

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

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

```
} return 0;
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 19

Section 1 : MCQ

1. 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

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

Answer

First In First Out

Status : Correct

Marks : 1/1

3. 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

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

Answer

underflow

Status : Wrong

Marks : 0/1

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

Answer

All of the mentioned options

Status : Correct

Marks : 1/1

6. 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

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

Answer

Enqueue and Dequeue

Status : Correct

Marks : 1/1

8. 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

9. 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

10. 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

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

Answer

Rear = MAX_SIZE - 1

Status : Correct

Marks : 1/1

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

Answer

Queue

Status : Correct

Marks : 1/1

13. 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

14. 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(cur);size--;return e;}}

```

Status : Correct

Marks : 1/1

15. 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

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. 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

10 30 40 15

Status : Correct

Marks : 1/1

18. What are the applications of dequeue?

Answer

All the mentioned options

Status : Correct

Marks : 1/1

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

Answer

Overflow

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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Customers can place orders and enjoy their delicious coffee drinks.

Input Format

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

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

Choice 2: Dequeue a coffee order from the queue.

Choice 3: Display the orders in the queue.

Choice 4: Exit the program.

Output Format

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

If the choice is 1:

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

If the choice is 2:

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

If the choice is 3:

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

If the choice is 4:

1. Exit the program and print "Exiting program"

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

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

Sample Test Case

Input: 1 L

1 E

1 M

1 O

1 N

1 O

3

2

3

4

Output: Order for L is enqueued.

Order for E is enqueued.

Order for M is enqueued.

Order for O is enqueued.

Order for N is enqueued.

Queue is full. Cannot enqueue more orders.

Orders in the queue are: L E M O N

Dequeued Order: L

Orders in the queue are: E M O N

Exiting program

Answer

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

```
#define MAX_SIZE 5
```

```
char orders[MAX_SIZE];
```

```
int front = -1;
```

```
int rear = -1;
```

```
void initializeQueue() {
```

```
    front = -1;
```

```
    rear = -1;
```

```
}
```

```
// You are using GCC
```

```
int isEmpty() {  
    //Type your code here  
    return (rear == -1 || rear < front + 1);  
};
```

```
int isFull() {  
    //Type your code here  
    return (rear - front == MAX_SIZE);  
};
```

```
int enqueue(char order) {  
    //Type your code here  
    if(isFull()){  
        printf("Queue is full. Cannot enqueue more orders.\n");  
        return 0;  
    }  
    else{  
        orders[++rear] = order;  
        printf("Order for %c is enqueued.\n", order);  
        return 1;  
    }  
}
```

```
int dequeue() {  
    //Type your code here  
    if(isEmpty()){  
        printf("No orders in the queue.\n");  
        return -1;  
    }  
    else{  
        printf("Dequeued Order: %c\n", orders[++front]);  
        return 1;  
    }  
}
```

```
void display() {  
    //Type your code here  
    if(isEmpty()){  
        printf("Queue is empty. No orders available.\n");  
        return;  
    }  
    else{
```

```

        printf("Orders in the queue are: ");
        for(int i = front+1; i<=rear; i++){
            printf("%c ", orders[i]);
        }
        printf("\n");
    }
}

```

```

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

```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Implement a program that provides the following functionalities:

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

helpdesk tickets currently in the queue.

Input Format

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

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

Choice 2: Dequeue a ticket from the queue.

Choice 3: Display the ticket IDs in the queue.

Choice 4: Exit the program.

Output Format

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

If the choice is 1:

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

If the choice is 2:

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

If the choice is 3:

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

If the choice is 4:

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1 101

1 202

1 203

1 204

1 205

1 206

3

2

3

4

Output: Helpdesk Ticket ID 101 is enqueued.

Helpdesk Ticket ID 202 is enqueued.

Helpdesk Ticket ID 203 is enqueued.

Helpdesk Ticket ID 204 is enqueued.

Helpdesk Ticket ID 205 is enqueued.

Queue is full. Cannot enqueue.

Helpdesk Ticket IDs in the queue are: 101 202 203 204 205

Dequeued Helpdesk Ticket ID: 101

Helpdesk Ticket IDs in the queue are: 202 203 204 205

Exiting the program

Answer

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

```
#define MAX_SIZE 5
```

```
int ticketIDs[MAX_SIZE];
```

```
int front = -1;
```

```
int rear = -1;
```

```
int lastDequeued;
```

```
void initializeQueue() {
```

```
    front = -1;
```

```
    rear = -1;
```

```
}
```

```
// You are using GCC
// lastDequeued = ticketIDs[rear];
//int lastDequeued;
int isEmpty() {
    //type your code here
    return (rear == -1 || rear<front);
}

int isFull() {
    //type your code here
    return rear == MAX_SIZE-1;
}

int enqueue(int ticketID) {
    //type your code here
    if(isFull()){
        printf("Queue is full. Cannot enqueue.\n");
        return 0;
    }
    else{
        if(front == -1){
            front++;
            //return 0;
        }

        ticketIDs[++rear] = ticketID;
        printf("Helpdesk Ticket ID %d is enqueued.\n", ticketID);
        return 0;
    }
}
```

```
int dequeue(){
    //type your code here
    if(isEmpty()){
        return 0;
    }
    else{
        lastDequeued = ticketIDs[front];
        front++;
        return 1; //ticketIDs[++front];
    }
}
```



```
}
```

```
void display() {  
    //type your code here  
    if(isEmpty()){  
        printf("Queue is empty.\n");  
        return;  
    }  
    else{  
        printf("Helpdesk Ticket IDs in the queue are: ");  
        for(int i = front; i<= rear; i++){  
            printf("%d ", ticketIDs[i]);  
        }  
        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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

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

Input Format

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

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

Option 2: Delete an element from the queue.

Option 3: Display the elements in the queue.

Output Format

For option 1 (insertion):-

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

For option 2 (deletion):-

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

For option 3 (display):-

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

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1 10

3

5

Output: 10 is inserted in the queue.

Elements in the queue are: 10

Invalid option.

Answer

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

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

```
#define max 5
```

```
int queue[max];
```

```
int front = -1, rear = -1;
```

```
// You are using GCC
```

```
int insertq(int *data)
```

```
{
```

```
    //Type your code here
```

```
    if(rear == max-1 || rear - front == max-1){
```

```
        return 0;
```

```
    }
```

```
    else{
```

```
        if(front == -1){
```

```
            front++;
```

```
        }
```

```
        queue[++rear] = *data;
```

```
        return 1;
```

```
    }
```

```
}
```

```
int delq()
```

```
{
```

```
    //Type your code here
```

```
    if(rear == -1 || rear < front){
```

```
        printf("Queue is empty.\n");
```

```
        return 0;
```

```
    }
```

```
    else{
```

```
        printf("Deleted number is: %d\n", queue[front]);
```

```
        front ++;
```

```
        return 1;
```

```

    }
}

void display()
{
    //Type your code here
    if(rear == -1 || rear < front){
        printf("Queue is empty.\n");
    }
    else{
        printf("Elements in the queue are: ");
        for(int i = front; i<= rear; i++){
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
}

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

```

```
        printf("Invalid option.\n");  
        break;  
    }  
    }  
    return 0;  
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

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

Input Format

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

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

Output Format

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

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

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5

12 56 87 23 45

Output: Front: 12, Rear: 45

Performing Dequeue Operation:

Front: 56, Rear: 45

Answer

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

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

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

```
// You are using GCC
struct Node* queue = NULL;
void enqueue(int d) {
    //Type your code here
```

```

    struct Node* newnode = (struct Node*)malloc(sizeof(struct Node));
    newnode -> data = d;
    newnode -> next = NULL;
    if(queue == NULL){
        front = rear = newnode;
        queue = newnode;
        return;
    }
    else{
        rear -> next = newnode;
        rear = newnode;
        return;
    }
}

void printFrontRear() {
    //Type your code here
    printf("Front: %d, Rear: %d\n", front->data, rear->data);
    return;
}

void dequeue() {
    //Type your code here
    struct Node* temp = front;
    front = front->next;
    free(temp);
    return;
    // printf("performing Dequeue Operation;");
}

```

```

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: R.R CHANDRU

Email: 240701087@rajalakshmi.edu.in

Roll no: 240701087

Phone: 9003697366

Branch: REC

Department: I CSE AG

Batch: 2028

Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_PAH

Attempt : 1

Total Mark : 50

Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

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

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

Example

Input:

5

10 2 30 4 50

5

Output:

Original Queue: 10 2 30 4 50

Queue after selective dequeue: 2 4

Explanation:

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

Input Format

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

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

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

Output Format

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

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5

10 2 30 4 50

5

Output: Original Queue: 10 2 30 4 50

Queue after selective dequeue: 2 4

Answer

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

```
int main() {
    int n, multiple;
```

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

```
    int queue[n];
```

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

```
    scanf("%d", &multiple);
```

```
    printf("Original Queue: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");
```

```
    int newQueue[n];
    int newSize = 0;
```

```
    for (int i = 0; i < n; i++) {
        if (queue[i] % multiple != 0) {
            newQueue[newSize++] = queue[i];
        }
    }
```

```
    printf("Queue after selective dequeue: ");
    for (int i = 0; i < newSize; i++) {
        printf("%d ", newQueue[i]);
    }
```

```
}  
printf("\n");  
  
return 0;  
}
```

Status : Correct

Marks : 10/10

2. Problem Statement

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

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

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

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

Input Format

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

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

Output Format

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

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

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

Sample Test Case

Input: 6

14 52 63 95 68 49

Output: Queue: 14 52 63 95 68 49

Queue After Dequeue: 52 63 95 68 49

Answer

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

```
int main() {
    int n;
```

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

```
    int ticketQueue[n];
```

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

```
    printf("Queue: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", ticketQueue[i]);
    }
    printf("\n");
```

```
    if (n > 0) {
```



```

printf("Queue After Dequeue: ");
for (int i = 1; i < n; i++) {
    printf("%d ", ticketQueue[i]);
}
printf("\n");
} else {
    printf("Queue After Dequeue: \n");
}

return 0;
}

```

Status : Correct

Marks : 10/10

3. Problem Statement

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

Use an array for implementation.

Input Format

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

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

Output Format

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

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

101 102 103 104 105

Output: 102 103 104 105

Answer

// You are using GCC

#include <stdio.h>

#define MAX_CUSTOMERS 25

```
int main() {
    int queue[MAX_CUSTOMERS];
    int front = 0;
    int rear = -1;
    int n;
```

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

```
    if (n > 0) {
        for (int i = 0; i < n && i < MAX_CUSTOMERS; i++) {
            rear++;
            queue[rear] = 0;
            scanf("%d", &queue[rear]);
        }
    }
```

```
    if (rear < front) {

        printf("Underflow\n");
        printf("Queue is empty\n");
    } else {

        front++;
```

```

    if (rear < front) {
        printf("Queue is empty\n");
    } else {

        for (int i = front; i <= rear; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
}

return 0;
}

```

Status : Correct

Marks : 10/10

4. Problem Statement

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

Assist her in the program.

Input Format

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

The second line consists of N space-separated integers.

The third line consists of an integer K.

Output Format

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

23 45 93 87 25

4

Output: Enqueued: 23

Enqueued: 45

Enqueued: 93

Enqueued: 87

Enqueued: 25

The 4th largest element: 25

Answer

```
// You are using GCC
```

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

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

```
#define MAX_SIZE 10
```

```
void enqueue(int queue[], int *size, int value) {
```

```
    if (*size < MAX_SIZE) {
```

```
        queue[*size] = value;
```

```
        (*size)++;
```

```
        printf("Enqueued: %d\n", value);
```

```
    } else {
```

```
        printf("Queue is full\n");
```

```
    }
```

```
}
```

```
int kthLargest(int queue[], int size, int k) {
```

```
    int *sorted = (int *)malloc(size * sizeof(int));
```

```
    for (int i = 0; i < size; i++) {
```

```
        sorted[i] = queue[i];
```

```
    }
```

```
    for (int i = 0; i < size - 1; i++) {
```

```

        for (int j = 0; j < size - i - 1; j++) {
            if (sorted[j] < sorted[j + 1]) {

                int temp = sorted[j];
                sorted[j] = sorted[j + 1];
                sorted[j + 1] = temp;
            }
        }
    }

    int kthLargestElement = sorted[k - 1];
    free(sorted);
    return kthLargestElement;
}

int main() {
    int queue[MAX_SIZE];
    int size = 0;
    int N, K;

    scanf("%d", &N);

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

    scanf("%d", &K);

    int kthLargestElement = kthLargest(queue, size, K);
    printf("The %dth largest element: %d\n", K, kthLargestElement);

    return 0;
}

```

Status : Correct

Marks : 10/10

5. Problem Statement

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

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

Input Format

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

Enter -1 to terminate input.

Output Format

The output prints the space-separated dequeued elements.

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

Sample Test Case

Input: 1

2

3

4

-1

Output: Dequeued elements: 1 2 3 4

Answer

```
// You are using GCC
```

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

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

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

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

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

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

```
void enqueue(struct Queue* queue, int data) {  
  
    struct Node* newNode = createNode(data);
```

```
    if (queue->rear == NULL) {  
        queue->front = queue->rear = newNode;  
        return;  
    }
```

```
    queue->rear->next = newNode;  
    queue->rear = newNode;  
}
```

```
void dequeue(struct Queue* queue) {  
    if (queue->front == NULL) {  
        printf("Dequeued elements: ");
```

```
    return;  
}
```

```
struct Node* temp = queue->front;  
queue->front = queue->front->next;
```

```
printf("%d ", temp->data);
```

```
free(temp);
```

```
if (queue->front == NULL) {  
    queue->rear = NULL;  
}  
}
```

```
int main() {  
    struct Queue* queue = createQueue();  
    int value;
```

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

```
        enqueue(queue, value);  
    }
```

```
    printf("Dequeued elements: ");  
    while (queue->front != NULL) {  
        dequeue(queue);  
    }  
    printf("\n");
```

```
    free(queue);
```



```
} return 0;
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 13

Section 1 : MCQ

1. 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

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 pre-order traversal of a binary search tree with nodes: 50, 30, 20, 55, 32, 52, 57?

Answer

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

Status : Wrong

Marks : 0/1

4. 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

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

Answer

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

Status : Correct

Marks : 1/1

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

Answer

14

Status : Correct

Marks : 1/1

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

Answer

1, 2, 4, 13, 14, 18

Status : Correct

Marks : 1/1

8. 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, 30, 32, 57, 52, 55, 50

Status : Wrong

Marks : 0/1

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 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

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

Answer

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

Status : Correct

Marks : 1/1

12. 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

13. 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

14. 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

15. 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

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

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

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

Output Format

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 2 7
15

Output: 2 5 7 10

Answer

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

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

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

```
struct TreeNode* insert(struct TreeNode* root, int key) {
    struct TreeNode *newnode=createNode(key);
    if(root==NULL){
```

```

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

```

```

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

```

```

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(key==root->data){
        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){
        return;
    }
    else{
        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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

Write a program to help him solve this program.

Input Format

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

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

3 1 5 2 4

Output: 3 1 2 5 4

Answer

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

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

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

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

```
}
```

```
void printPreorder(struct Node* node) {  
    if(node==NULL){  
        return;  
    }  
    else{  
        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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

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

Write a program to calculate this efficiently.

Input Format

The first line of input consists of an integer n, representing the number of nodes

in the binary search tree.

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

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7

8 3 10 1 6 14 23

6

Output: Value 6 is found in the tree.

Answer

```
// You are using GCC
```

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

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

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

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

```
Node* insert(Node* root, int key) {
    if (root == NULL)
        return createNode(key);

    if (key < root->key)
        root->left = insert(root->left, key);
    else if (key > root->key)
        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, value, key;
    Node* root = NULL;

    scanf("%d", &n);

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

```
    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: R.R CHANDRU

Email: 240701087@rajalakshmi.edu.in

Roll no: 240701087

Phone: 9003697366

Branch: REC

Department: I CSE AG

Batch: 2028

Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 4

Attempt : 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){  
        return createNode(data);  
    }  
    else if(root->data>data){  
        root->left=insert(root->left,data);  
    }  
    else if(root->data<data){
```

```

        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 *temp=root;
    while(temp->left!=NULL){
        temp=temp->left;
    }
    return temp->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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

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

Output Format

The output prints the maximum value in the BST.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

10 5 15 2 7

Output: 15

Answer

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

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

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

// You are using GCC
struct TreeNode* insert(struct TreeNode* root, int key) {
    if(root==NULL){
        return createNode(key);
    }
    else if(key>root->data){
        root->right=insert(root->right,key);
    }
    else if(key<root->data){
        root->left=insert(root->left,key);
    }
    return root;
}
```

```
int findMax(struct TreeNode* root) {  
    if(root->right==NULL){  
        return root->data;  
    }  
    return findMax(root->right);  
}
```

```
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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_MCQ_Updated_1

Attempt : 1
Total Mark : 20
Marks Obtained : 20

Section 1 : MCQ

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

Answer

Two sorted subarrays are combined into one sorted array

Status : Correct

Marks : 1/1

2. 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

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

Answer

Quicksort requires less auxiliary space

Status : Correct

Marks : 1/1

4. Which of the following sorting algorithms is based on the divide and conquer method?

Answer

Merge Sort

Status : Correct

Marks : 1/1

5. Is Merge Sort a stable sorting algorithm?

Answer

Yes, always stable.

Status : Correct

Marks : 1/1

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

Answer

It requires additional memory for merging

Status : Correct

Marks : 1/1

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

Answer

It can be implemented as a stable sort

Status : Correct

Marks : 1/1

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

Answer

When sorting linked lists

Status : Correct

Marks : 1/1

9. 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

10. 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

11. 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

12. 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

13. 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

14. Merge sort is _____.

Answer

Comparison-based sorting algorithm

Status : Correct

Marks : 1/1

15. Which of the following is true about Quicksort?

Answer

It is an in-place sorting algorithm

Status : Correct

Marks : 1/1

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

Answer

merging

Status : Correct

Marks : 1/1

17. 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

18. Let P be a quick sort program to sort numbers in ascending order using the first element as a pivot. Let t_1 and t_2 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

$t_1 > t_2$

Status : Correct

Marks : 1/1

19. 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

20. 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

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 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 merged[], int arr1[], int arr2[], int n1, int n2) {  
    int i = 0, j = 0, k = 0;  
    while(i < n1 && j < n2) {  
        if(arr1[i] < arr2[j]) {  
            merged[k++] = arr1[i++];  
        } else {  
            merged[k++] = arr2[j++];  
        }  
    }  
    while(i < n1) {  
        merged[k++] = arr1[i++];  
    }  
    while(j < n2) {  
        merged[k++] = arr2[j++];  
    }  
}
```

```
void mergeSort(int arr[], int n) {  
    if(n <= 1) return;
```

```
int mid = n / 2;  
mergeSort(arr, mid);
```

```
mergeSort(arr + mid, n - mid);  
int temp[n];  
int i = 0, j = mid, k = 0;
```

```
while(i < mid && j < n) {  
    if(arr[i] < arr[j]) {  
        temp[k++] = arr[i++];  
    } else {  
        temp[k++] = arr[j++];  
    }  
}
```

```
while(i < mid) {  
    temp[k++] = arr[i++];  
}
```

```
while(j < n) {  
    temp[k++] = arr[j++];  
}
```

```
for(int p = 0; p < n; p++) {  
    arr[p] = temp[p];  
}
```

```
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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 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]);  
    }  
}
```

```
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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 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: R.R CHANDRU

Email: 240701087@rajalakshmi.edu.in

Roll no: 240701087

Phone: 9003697366

Branch: REC

Department: I CSE AG

Batch: 2028

Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 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 n th 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 n th 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>
```

```
// You are using GCC
```

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

```
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++;
            swap(&arr[i], &arr[j]);
```

```
        }
    }
    swap(&arr[i + 1], &arr[high]);
```

```
        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 arr[], int n, int k) {
    quickSort(arr, 0, n - 1);
    printf("%d\n", arr[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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 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>
```

```
void merge(double arr[], int left, int mid, int right) {
```

```
    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;
```

```
    double L[n1], R[n2];
```

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

```
        L[i] = arr[left + i];
```

```
    for (j = 0; j < n2; j++)
```

```
        R[j] = arr[mid + 1 + j];
```

```
    i = 0, j = 0, k = left;
```

```
    while (i < n1 && j < n2) {
```

```
        if (L[i] <= R[j])
```

```
            arr[k++] = L[i++];
```

```
        else
```

```
            arr[k++] = R[j++];
```

```
    }
```

```

while (i < n1)
    arr[k++] = L[i++];
while (j < n2)
    arr[k++] = R[j++];
}

void mergeSort(double arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

int main() {
    int n;
    scanf("%d", &n);
    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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 20

Section 1 : MCQ

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

Answer

$$h(k) = k \% m$$

Status : Correct

Marks : 1/1

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

Answer

A prime number

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

Answer

$(i + 1) \% \text{table_size}$

Status : Correct

Marks : 1/1

5. 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

6. 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

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

Answer

8

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. 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

10. 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

11. Which situation causes clustering in linear probing?

Answer

All the mentioned options

Status : Correct

Marks : 1/1

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

Answer

$k \% \text{table_size}$

Status : Correct

Marks : 1/1

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

Answer

index = (index + 1) % size;

Status : Correct

Marks : 1/1

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

Answer

A special "deleted" marker

Status : Correct

Marks : 1/1

15. 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

16. 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

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

Answer

Array

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

Answer

Mid-Square

Status : Correct

Marks : 1/1

20. What is the primary disadvantage of linear probing?

Answer

Clustering

Status : Correct

Marks : 1/1

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 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: $\text{index} = \text{roll_number} \% \text{table_size}$ On 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
```

```
// You are using GCC
```

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

```
void insertIntoHashTable(int table[], int table_size, int arr[], int n) {  
    for (int i = 0; i < n; i++) {  
        int index = arr[i] % table_size;  
  
        while (table[index] != -1) {
```

```
        index = (index + 1) % table_size;
    }
    table[index] = arr[i];
}
}
```

```
void printTable(int table[], int table_size) {
    for (int i = 0; i < table_size; i++) {
        printf("%d ", table[i]);
    }
}
```

```
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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 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
```

```
// You are using GCC
```

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

```
void insertIntoHashTable(int table[], int table_size, int arr[], int n) {  
    for (int i = 0; i < n; i++) {
```

```

    int index = arr[i] % table_size;

    while (table[index] != -1) {
        index = (index + 1) % table_size;
    }
    table[index] = arr[i];
}
}

```

```

int searchInHashTable(int table[], int table_size, int x) {
    int index = x % table_size;
    int startIndex = index;

    while (table[index] != -1) {
        if (table[index] == x) {
            return 1;
        }
        index = (index + 1) % table_size;
        if (index == startIndex) {
            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: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 7.5

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

```
int hash(const char *key) {  
    int hash = 0;  
    for (int i = 0; key[i] != '\0'; i++) {  
        hash += key[i];  
    }  
}
```

```

    return hash;
}

void insertKeyValuePair(Dictionary *dict, const char *key, const char *value) {
    if (dict->size == dict->capacity) {
        dict->capacity *= 2;
        dict->pairs = (KeyValuePair *)realloc(dict->pairs, dict->capacity *
sizeof(KeyValuePair));
    }
    strcpy(dict->pairs[dict->size].key, key);
    strcpy(dict->pairs[dict->size].value, value);
    dict->size++;
}

int doesKeyExist(Dictionary *dict, const char *key) {
    for (int i = 0; i < dict->size; i++) {
        if (strcmp(dict->pairs[i].key, key) == 0) {
            return i;
        }
    }
    return -1;
}

void removeKeyValuePair(Dictionary *dict, const char *key) {
    int index = doesKeyExist(dict, key);
    if (index != -1) {
        for (int i = index; i < dict->size - 1; i++) {
            strcpy(dict->pairs[i].key, dict->pairs[i + 1].key);
            strcpy(dict->pairs[i].value, dict->pairs[i + 1].value);
        }
        dict->size--;
    }
}

void printDictionary(Dictionary *dict) {
    for (int i = 0; i < dict->size; i++) {
        printf("Key: %s; Value: %s\n", dict->pairs[i].key, dict->pairs[i].value);
    }
}

```

Status : Partially correct

Marks : 7.5/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 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

```
int keyExists(KeyValuePair* dictionary, int size, const char* key) {  
    for (int i = 0; i < size; i++) {  
        if (strcmp(dictionary[i].key, key) == 0) {  
            return 1;  
        }  
    }  
    return 0;  
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: R.R CHANDRU
Email: 240701087@rajalakshmi.edu.in
Roll no: 240701087
Phone: 9003697366
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 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 -> $\text{hash}(2*2) \% 100 = 4$

3 -> $\text{hash}(3*3) \% 100 = 9$

4 -> $\text{hash}(4*4) \% 100 = 16$

5 -> $\text{hash}(5*5) \% 100 = 25$

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 squared = key * key;
    int middle = (squared / 10) % 100; // Extract middle 2 digits
    return middle % tableSize;
}

int getOddOccurrence(int arr[], int size) {
    int hashTable[MAX_SIZE];
    int countTable[MAX_SIZE];
    memset(hashTable, 0, sizeof(hashTable));
    memset(countTable, 0, sizeof(countTable));

    for (int i = 0; i < size; i++) {
        int h = hash(arr[i], MAX_SIZE);
        bool found = false;
        for (int j = 0; j < MAX_SIZE; j++) {
            int idx = (h + j) % MAX_SIZE;
            if (countTable[idx] == 0) {
                hashTable[idx] = arr[i];
                countTable[idx] = 1;
            }
        }
    }
}
```

```

        break;
    } else if (hashTable[idx] == arr[i]) {
        countTable[idx]++;
        found = true;
        break;
    }
}
}

for (int i = 0; i < MAX_SIZE; i++) {
    if (countTable[i] % 2 == 1) {
        return hashTable[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