

**NAGARJUNA COLLEGE OF ENGINEERING AND TECHNOLOGY**

**Department of CSE (Data Science)**

**DS LAB USING C**

**MANUAL**

**22CDL35**

**PREPARED BY**

R.RADHAKRISHNAN.,M.E.,(PhD).,

ASSISTANT PROFESSOR

NIKHILA M.R.,M.E.,

ASSISTANT PROFESSOR

Department of CSE (Data Science)

1. **Design, Develop and Implement a menu driven Program in C for the following Array Operations**
2. **Creating an Array of N Integer Elements**
3. **Display of Array Elements with Suitable Headings**
4. **Exit.**

**Support the program with functions for each of the above operations.**

#include<stdio.h>

#include<stdlib.h>

int a[20];

int n,val,i,pos;

/\*Function Prototype\*/

void create();

void display();

int main()

{

int choice=1;

while(choice)

{

printf("\n\n--------MENU-----------\n");

printf("1.CREATE\n");

printf("2.DISPLAY\n");

printf("3.EXIT\n");

printf("Enter Your Choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

create();

break;

case 2:

display();

break;

case 3:

exit(0);

break;

default:

printf("\nInvalid choice:\n");

}

}

return 0;

}

void create()

{

printf("\nEnter the size of the array elements:\t");

scanf("%d",&n);

printf("\nEnter the elements for the array:\n");

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

{

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

}

}

//displaying an array elements

void display()

{

int i;

printf("\nThe array elements are:\n");

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

{

printf("%d\t",a[i]);

}

}

Sample O/P

--------MENU-----------

1.CREATE

2.DISPLAY

3.EXIT

Enter Your Choice

1

Enter the size of the array elements: 2

Enter the elements for the array:

22

33

--------MENU-----------

1.CREATE

2.DISPLAY

3.EXIT

Enter Your Choice

2

The array elements are:

22 33

--------MENU-----------

1.CREATE

2.DISPLAY

3.EXIT

Enter Your Choice

3

**2. Design, Develop and Implement a menu driven Program in C for the following**

**Array operations**

**a. Inserting an Element (ELEM) at a given valid Position (POS)**

**b. Deleting an Element at a given valid Position POS)**

**c. Display of Array Elements Exit.**

**d. Support the program with functions for each of the above operations**

#include<stdio.h>

#include<stdlib.h>

int a[20];

int n,val,i,pos;

/\*Function Prototype\*/

void create();

void display();

void insert();

void delete();

int main()

{

int choice=1;

while(choice)

{

printf("\n\n--------MENU-----------\n");

printf("1.CREATE\n");

printf("2.DISPLAY\n");

printf("3.INSERT\n");

printf("4.DELETE\n");

printf("5.EXIT\n");

printf("-----------------------");

printf("\nENTER YOUR CHOICE:\t");

scanf("%d",&choice);

switch(choice)

{

case 1:

create();

break;

case 2:

display();

break;

case 3:

insert();

break;

case 4:

delete();

break;

case 5:

exit(0);

default:

printf("\nInvalid choice:\n");

break;

}

}

return 0;

}

//creating an array

void create()

{

printf("\nEnter the size of the array elements:\t");

scanf("%d",&n);

printf("\nEnter the elements for the array:\n");

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

{

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

}

}

//displaying an array elements

void display()

{

int i;

printf("\nThe array elements are:\n");

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

{

printf("%d\t",a[i]);

}

}

//inserting an element into an array

void insert()

{

printf("\nEnter the position for the new element:\t")

scanf("%d",&pos);

printf("\nEnter the element to be inserted :\t");

scanf("%d",&val);

for(i=n-1;i>=pos;i--)

{

a[i+1]=a[i];

}

a[pos]=val;

n=n+1;

}

//deleting an array element

void delete()

{

printf("\nEnter the position of the element to be deleted:\t");

scanf("%d",&pos);

val=a[pos];

for(i=pos;i<n-1;i++)

{

a[i]=a[i+1];

}

n=n-1;

printf("\nThe deleted element is =%d",val);

}

**Output**

--------MENU-----------

1.CREATE

2.DISPLAY

3.INSERT

4.DELETE

5.EXIT

-----------------------

ENTER YOUR CHOICE: 1

Enter the size of the array elements: 3

Enter the elements for the array

10 25 30

ENTER YOUR CHOICE: 2 The

array elements are:

10 25 30

ENTER YOUR CHOICE: 3

Enter the position for the new element: 1

Enter the element to be inserted : 20

ENTER YOUR CHOICE: 2 The

array elements are:

10 20 25 30

ENTER YOUR CHOICE: 4

Enter the position of the element to be deleted: 3

The deleted element is =30

enter your choice: 5

**3.Design, Develop and Implement a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)**

**a. Push an Element on to Stack**

**b. Pop an Element from Stack**

**c. Demonstrate Overflow and Underflow situations on Stack**

**d. Display the status of Stack**

**e. Exit Support the program with appropriate functions for each of the above**

**operations**

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

#define max\_size 5

int stack[max\_size],top = -1;

void push();

void pop();

void display();

int main()

{

int choice=1;

while(choice)

{

printf("\n\n--------STACK OPERATIONS-----------\n");

printf("1.Push\n");

printf("2.Pop\n");

printf("3.Display\n");

printf("4.Exit\n");

printf("-----------------------");

printf("\nEnter your choice:\t");

scanf("%d",&choice);

switch(choice)

{

case 1:

push();

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

printf("\nInvalid choice:\n");

break;

}

}

return 0;

}

void push() //Inserting element into the stack

{

int item,n;

if(top==(max\_size-1))

{

printf("\nStack Overflow:");

}

else

{

printf("Enter the element to be inserted:\t");

scanf("%d",&item);

top=top+1;

stack[top]=item;

}

}

void pop() //deleting an element from the stack

{

int item;

if(top==-1)

{

printf("Stack Underflow:");

}

else

{

item=stack[top];

top=top-1;

printf("\nThe poped element: %d\t",item);

}

**}**

void display()

{

int i;

if(top==-1)

{

printf("\nStack is Empty:");

}

else

{

printf("\nThe stack elements are:\n" );

for(i=top;i>=0;i--)

{

printf("%d\n",stack[i]);

}

}

}

**Output**

--------STACK OPERATIONS-----------

1.Push

2.Pop

3.Display

4.Exit

-----------------------

Enter your choice: 1

Enter the element to be inserted: 1

Enter your choice: 1

Enter the element to be inserted: 2

Enter your choice: 1

Enter the element to be inserted: 1

Enter your choice: 1

Enter the element to be inserted: 5

Enter your choice: 2

The poped element: 5

Enter your choice: 4 The stack elements are:

1 2 1

1. **Design, Develop and Implement a Program in C for the Evaluation of Suffix expression with single digit operands and operators: +, -, \*, /, %, ^.**

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <math.h>

#define MAX\_SIZE 100

typedef struct {

int top;

int stack[MAX\_SIZE];

} Stack;

void initialize(Stack \*s) {

s->top = -1;

}

void push(Stack \*s, int value) {

s->stack[++s->top] = value;

}

int pop(Stack \*s) {

return s->stack[s->top--];

}

int isOperator(char ch) {

return (ch == '+' || ch == '-' || ch == '\*' || ch == '/' || ch == '%' || ch == '^');

}

int evaluatePostfix(char postfix[]) {

Stack stack;

initialize(&stack);

for (int i = 0; postfix[i] != '\0'; i++) {

if (isdigit(postfix[i])) {

push(&stack, postfix[i] - '0');

} else if (isOperator(postfix[i])) {

int operand2 = pop(&stack);

int operand1 = pop(&stack);

int result;

switch (postfix[i]) {

case '+': result = operand1 + operand2; break;

case '-': result = operand1 - operand2; break;

case '\*': result = operand1 \* operand2; break;

case '/': result = operand1 / operand2; break;

case '%': result = operand1 % operand2; break;

case '^': result = (int)pow(operand1, operand2); break;

}

push(&stack, result);

}

}

return pop(&stack);

}

int main() {

char postfix[MAX\_SIZE];

printf("Enter postfix expression: ");

scanf("%s", postfix);

int result = evaluatePostfix(postfix);

printf("Result: %d\n", result);

return 0;

}

**Output**

Enter postfix expression: 76/

Result: 1

Enter postfix expression: 45\*

Result: 20

1. **Design, Develop and Implement a Program in C for Solving Tower of Hanoi problem with n disks.**

#include <stdio.h>

void towerOfHanoi(int n, char source, char intermediate, char destination) {

if (n == 1) {

printf("Move disk 1 from %c to %c\n", source, destination);

return;

}

towerOfHanoi(n - 1, source, destination, intermediate);

printf("Move disk %d from %c to %c\n", n, source, destination);

towerOfHanoi(n - 1, intermediate, source, destination);

}

int main() {

int n;

// Input number of disks

printf("Enter the number of disks: ");

scanf("%d", &n);

// Towers are represented as 'A', 'B', and 'C'

towerOfHanoi(n, 'A', 'B', 'C');

return 0;

}

Sample O/P

Enter the number of disks: 3

Move disk 1 from A to C

Move disk 2 from A to B

Move disk 1 from C to B

Move disk 3 from A to C

Move disk 1 from B to A

Move disk 2 from B to C

Move disk 1 from A to C

1. Implement a Menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Avg\_Marks

a. Create N number of Students Data by inserting at end of the list.

b. Insert and Delete at front of the list

c. Delete at the end of list

d. Display the status of SLL

d. Demonstration stack and queue

e. Exit

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct Student {

char usn[20];

char name[30];

float avg\_marks;

struct Student\* next;

};

struct Student\* createStudent() {

struct Student\* newStudent = (struct Student\*)malloc(sizeof(struct Student));

if (newStudent == NULL) {

printf("Memory allocation failed.\n");

exit(1);

}

printf("Enter USN: ");

scanf("%s", newStudent->usn);

printf("Enter Name: ");

scanf("%s", newStudent->name);

printf("Enter Average Marks: ");

scanf("%f", &newStudent->avg\_marks);

newStudent->next = NULL;

return newStudent;

}

void insertAtEnd(struct Student\*\* head) {

struct Student\* newStudent = createStudent();

if (\*head == NULL) {

\*head = newStudent;

} else {

struct Student\* temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newStudent;

}

printf("Student data inserted at the end successfully.\n");

}

void insertAtFront(struct Student\*\* head) {

struct Student\* newStudent = createStudent();

newStudent->next = \*head;

\*head = newStudent;

printf("Student data inserted at the front successfully.\n");

}

void deleteAtFront(struct Student\*\* head) {

if (\*head == NULL) {

printf("Stack is empty. Unable to pop.\n");

return;

}

struct Student\* temp = \*head;

\*head = (\*head)->next;

free(temp);

printf("Student data deleted from the front successfully.\n");

}

void displayList(struct Student\* head) {

printf("\nStudent List:\n");

while (head != NULL) {

printf("USN: %s, Name: %s, Avg Marks: %.2f\n", head->usn, head->name, head->avg\_marks);

head = head->next;

}

printf("\n");

}

void demonstrateStack(struct Student\*\* head) {

printf("Demonstrating Stack using SLL:\n");

insertAtFront(head);

insertAtFront(head);

insertAtFront(head);

displayList(\*head);

deleteAtFront(head);

displayList(\*head);

}

void demonstrateQueue(struct Student\*\* head) {

printf("Demonstrating Queue using SLL:\n");

insertAtEnd(head);

insertAtEnd(head);

insertAtEnd(head);

displayList(\*head);

deleteAtFront(head);

displayList(\*head);

}

int main() {

struct Student\* head = NULL;

int choice;

do {

printf("\nMenu:\n");

printf("1. Insert at the end\n");

printf("2. Insert at the front\n");

printf("3. Delete at the front (Stack pop)\n");

printf("4. Display status\n");

printf("5. Demonstrate Stack\n");

printf("6. Demonstrate Queue\n");

printf("7. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

insertAtEnd(&head);

break;

case 2:

insertAtFront(&head);

break;

case 3:

deleteAtFront(&head);

break;

case 4:

displayList(head);

break;

case 5:

demonstrateStack(&head);

break;

case 6:

demonstrateQueue(&head);

break;

case 7:

printf("Exiting the program.\n");

break;

default:

printf("Invalid choice. Please enter a valid option.\n");

}

} while (choice != 7);

return 0;

}

Sample o/p

Menu:

1. Insert at the end

2. Insert at the front

3. Delete at the front (Stack pop)

4. Display status

5. Demonstrate Stack

6. Demonstrate Queue

7. Exit

Enter your choice: 4

Student List:

Menu:

1. Insert at the end

2. Insert at the front

3. Delete at the front (Stack pop)

4. Display status

5. Demonstrate Stack

6. Demonstrate Queue

7. Exit

Enter your choice: 5

Demonstrating Stack using SLL:

Enter USN: 1

Enter Name: a1

Enter Average Marks: 22

Student data inserted at the front successfully.

Enter USN: 2

Enter Name: a2

Enter Average Marks: 44

Student data inserted at the front successfully.

Enter USN: 3

Enter Name: a3

Enter Average Marks: 55

Student data inserted at the front successfully.

Student List:

USN: 3, Name: a3, Avg Marks: 55.00

USN: 2, Name: a2, Avg Marks: 44.00

USN: 1, Name: a1, Avg Marks: 22.00

Student data deleted from the front successfully.

Student List:

USN: 2, Name: a2, Avg Marks: 44.00

USN: 1, Name: a1, Avg Marks: 22.00

Menu:

1. Insert at the end

2. Insert at the front

3. Delete at the front (Stack pop)

4. Display status

5. Demonstrate Stack

6. Demonstrate Queue

7. Exit

Enter your choice: 4

Student List:

USN: 2, Name: a2, Avg Marks: 44.00

USN: 1, Name: a1, Avg Marks: 22.00

Menu:

1. Insert at the end

2. Insert at the front

3. Delete at the front (Stack pop)

4. Display status

5. Demonstrate Stack

6. Demonstrate Queue

7. Exit

Enter your choice: 6

Demonstrating Queue using SLL:

Enter USN: 11

Enter Name: b1

Enter Average Marks: 66

Student data inserted at the end successfully.

Enter USN: 12

Enter Name: b2

Enter Average Marks: 67

Student data inserted at the end successfully.

Enter USN: 13

Enter Name: b3

Enter Average Marks: 68

Student data inserted at the end successfully.

Student List:

USN: 2, Name: a2, Avg Marks: 44.00

USN: 1, Name: a1, Avg Marks: 22.00

USN: 11, Name: b1, Avg Marks: 66.00

USN: 12, Name: b2, Avg Marks: 67.00

USN: 13, Name: b3, Avg Marks: 68.00

Student data deleted from the front successfully.

Student List:

USN: 1, Name: a1, Avg Marks: 22.00

USN: 11, Name: b1, Avg Marks: 66.00

USN: 12, Name: b2, Avg Marks: 67.00

USN: 13, Name: b3, Avg Marks: 68.00

1. Design and Develop following operations on Doubly Linked List (DLL) of Employee Data with the fields: *SSN, Name, Dept etc.*

a. Create a Node of N Employees Data by inserting in front.

b. Insert a new node to the right of key value.

c. Perform Insertion and Deletion at End of DLL

d. Display the status of DLL and count the number of nodes

e. Exit

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Employee structure

struct Employee {

char ssn[12];

char name[50];

char department[50];

struct Employee\* next;

struct Employee\* prev;

};

// Function to create a new employee node

struct Employee\* createEmployeeNode(char ssn[], char name[], char department[]) {

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

if (newNode == NULL) {

printf("Memory allocation failed\n");

exit(1);

}

strcpy(newNode->ssn, ssn);

strcpy(newNode->name, name);

strcpy(newNode->department, department);

newNode->next = NULL;

newNode->prev = NULL;

return newNode;

}

// Function to accept employee data

void acceptEmployeeData(char ssn[], char name[], char department[]) {

printf("Enter SSN of employee: ");

scanf("%s", ssn);

printf("Enter Name of employee: ");

scanf("%s", name);

printf("Enter Department of employee: ");

scanf("%s", department);

}

// Function to insert an employee at the beginning

void insertEmployeeAtBeginning(struct Employee\*\* head, char ssn[], char name[], char department[]) {

struct Employee\* newNode = createEmployeeNode(ssn, name, department);

if (\*head == NULL) {

\*head = newNode;

} else {

newNode->next = \*head;

(\*head)->prev = newNode;

\*head = newNode;

}

}

// Function to insert an employee to the right of the key value

void insertEmployeeAfterKey(struct Employee\*\* head, char ssn[], char name[], char department[], char keySSN[]) {

struct Employee\* newNode = createEmployeeNode(ssn, name, department);

struct Employee\* temp = \*head;

while (temp != NULL && strcmp(temp->ssn, keySSN) != 0) {

temp = temp->next;

}

if (temp == NULL) {

printf("Key value not found. Cannot insert.\n");

free(newNode);

return;

}

newNode->next = temp->next;

newNode->prev = temp;

if (temp->next != NULL) {

temp->next->prev = newNode;

}

temp->next = newNode;

printf("Employee inserted to the right of key value %s.\n", keySSN);

}

// Function to insert an employee at the end

void insertEmployeeAtEnd(struct Employee\*\* head, char ssn[], char name[], char department[]) {

struct Employee\* newNode = createEmployeeNode(ssn, name, department);

if (\*head == NULL) {

\*head = newNode;

} else {

struct Employee\* temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

newNode->prev = temp;

}

}

// Function to display the list of employees and count the number of nodes

void displayEmployeeList(struct Employee\* head) {

int count = 0;

printf("Employee List:\n");

while (head != NULL) {

printf("SSN: %s, Name: %s, Department: %s\n", head->ssn, head->name, head->department);

head = head->next;

count++;

}

printf("Number of nodes: %d\n", count);

}

// Function to delete an employee from the end

void deleteEmployeeAtEnd(struct Employee\*\* head) {

if (\*head == NULL) {

printf("List is empty. Cannot delete.\n");

return;

}

struct Employee\* temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

if (temp->prev != NULL) {

temp->prev->next = NULL;

} else {

\*head = NULL;

}

free(temp);

printf("Employee deleted from the end.\n");

}

// Function to free the memory allocated for the list of employees

void freeEmployeeList(struct Employee\* head) {

struct Employee\* temp;

while (head != NULL) {

temp = head;

head = head->next;

free(temp);

}

}

int main() {

struct Employee\* head = NULL;

int choice;

char ssn[12], name[50], department[50], keySSN[12];

do {

printf("\n----- MENU -----\n");

printf("1. Create a Node of N Employees Data by inserting in front\n");

printf("2. Insert a new node to the right of key value\n");

printf("3. Perform Insertion at End of DLL\n");

printf("4. Display the status of DLL and count the number of nodes\n");

printf("5. Delete Employee at End\n");

printf("6. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

acceptEmployeeData(ssn, name, department);

insertEmployeeAtBeginning(&head, ssn, name, department);

break;

case 2:

printf("Enter SSN of employee to the right of which new node should be inserted: ");

scanf("%s", keySSN);

acceptEmployeeData(ssn, name, department);

insertEmployeeAfterKey(&head, ssn, name, department, keySSN);

break;

case 3:

acceptEmployeeData(ssn, name, department);

insertEmployeeAtEnd(&head, ssn, name, department);

break;

case 4:

displayEmployeeList(head);

break;

case 5:

deleteEmployeeAtEnd(&head);

break;

case 6:

freeEmployeeList(head);

printf("Memory freed. Exiting...\n");

break;

default:

printf("Invalid choice. Try again.\n");

}

} while (choice != 6);

return 0;

}

SAMPLE O/P

----- MENU -----

1. Create a Node of N Employees Data by inserting in front

2. Insert a new node to the right of key value

3. Perform Insertion at End of DLL

4. Display the status of DLL and count the number of nodes

5. Delete Employee at End

6. Exit

Enter your choice: 1

Enter SSN of employee: 30

Enter Name of employee: c

Enter Department of employee: d3

----- MENU -----

1. Create a Node of N Employees Data by inserting in front

2. Insert a new node to the right of key value

3. Perform Insertion at End of DLL

4. Display the status of DLL and count the number of nodes

5. Delete Employee at End

6. Exit

Enter your choice: 1

Enter SSN of employee: 10

Enter Name of employee: A

Enter Department of employee: d1

----- MENU -----

1. Create a Node of N Employees Data by inserting in front

2. Insert a new node to the right of key value

3. Perform Insertion at End of DLL

4. Display the status of DLL and count the number of nodes

5. Delete Employee at End

6. Exit

Enter your choice: 4

Employee List:

SSN: 10, Name: A, Department: d1

SSN: 30, Name: c, Department: d3

Number of nodes: 2

----- MENU -----

1. Create a Node of N Employees Data by inserting in front

2. Insert a new node to the right of key value

3. Perform Insertion at End of DLL

4. Display the status of DLL and count the number of nodes

5. Delete Employee at End

6. Exit

Enter your choice: 2

Enter SSN of employee to the right of which new node should be inserted: 10

Enter SSN of employee: 20

Enter Name of employee: B

Enter Department of employee: d2

Employee inserted to the right of key value 10.

----- MENU -----

1. Create a Node of N Employees Data by inserting in front

2. Insert a new node to the right of key value

3. Perform Insertion at End of DLL

4. Display the status of DLL and count the number of nodes

5. Delete Employee at End

6. Exit

Enter your choice: 4

Employee List:

SSN: 10, Name: A, Department: d1

SSN: 20, Name: B, Department: d2

SSN: 30, Name: c, Department: d3

Number of nodes: 3

----- MENU -----

1. Create a Node of N Employees Data by inserting in front

2. Insert a new node to the right of key value

3. Perform Insertion at End of DLL

4. Display the status of DLL and count the number of nodes

5. Delete Employee at End

6. Exit

Enter your choice: 3

Enter SSN of employee: 40

Enter Name of employee: D

Enter Department of employee: d4

----- MENU -----

1. Create a Node of N Employees Data by inserting in front

2. Insert a new node to the right of key value

3. Perform Insertion at End of DLL

4. Display the status of DLL and count the number of nodes

5. Delete Employee at End

6. Exit

Enter your choice: 4

Employee List:

SSN: 10, Name: A, Department: d1

SSN: 20, Name: B, Department: d2

SSN: 30, Name: c, Department: d3

SSN: 40, Name: D, Department: d4

Number of nodes: 4

----- MENU -----

1. Create a Node of N Employees Data by inserting in front

2. Insert a new node to the right of key value

3. Perform Insertion at End of DLL

4. Display the status of DLL and count the number of nodes

5. Delete Employee at End

6. Exit

Enter your choice: 5

Employee deleted from the end.

----- MENU -----

1. Create a Node of N Employees Data by inserting in front

2. Insert a new node to the right of key value

3. Perform Insertion at End of DLL

4. Display the status of DLL and count the number of nodes

5. Delete Employee at End

6. Exit

Enter your choice: 4

Employee List:

SSN: 10, Name: A, Department: d1

SSN: 20, Name: B, Department: d2

SSN: 30, Name: c, Department: d3

Number of nodes: 3

1. Design, Develop and Implement a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)

i.Insert an Element on to Circular QUEUE

ii.Delete an Element from Circular QUEUE

iii.Demonstrate Overflow and Underflow situations on Circular QUEUE

iv.Display the status of Circular QUEUE`

v.Exit

Support the program with appropriate functions for each of the above operations

#include <stdio.h>

#include <stdlib.h>

#define MAX 5

// Circular Queue structure

struct CircularQueue {

char items[MAX];

int front, rear;

};

// Function prototypes

void initializeQueue(struct CircularQueue \*cq);

void insertElement(struct CircularQueue \*cq, char element);

char deleteElement(struct CircularQueue \*cq);

void displayQueue(struct CircularQueue cq);

int isFull(struct CircularQueue cq);

int isEmpty(struct CircularQueue cq);

int main() {

struct CircularQueue cq;

initializeQueue(&cq);

int choice;

char element;

do {

// Display menu

printf("\nCircular Queue Operations:\n");

printf("1. Insert Element\n");

printf("2. Delete Element\n");

printf("3. Demonstrate Overflow and Underflow\n");

printf("4. Display Queue\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the element to be inserted: ");

scanf(" %c", &element);

insertElement(&cq, element);

break;

case 2:

element = deleteElement(&cq);

if (element != '\0') {

printf("Deleted element: %c\n", element);

}

break;

case 3:

if (isFull(cq)) {

printf("Queue is Full (Overflow)\n");

} else if (isEmpty(cq)) {

printf("Queue is Empty (Underflow)\n");

} else {

printf("Queue has elements\n");

}

break;

case 4:

displayQueue(cq);

break;

case 5:

printf("Exiting program.\n");

break;

default:

printf("Invalid choice. Please enter a valid option.\n");

}

} while (choice != 5);

return 0;

}

// Function to initialize the Circular Queue

void initializeQueue(struct CircularQueue \*cq) {

cq->front = cq->rear = -1;

}

// Function to insert an element into the Circular Queue

void insertElement(struct CircularQueue \*cq, char element) {

if ((cq->front == 0 && cq->rear == MAX - 1) || (cq->rear == (cq->front - 1) % (MAX - 1))) {

printf("Queue is Full (Overflow)\n");

return;

}

if (cq->front == -1) {

cq->front = cq->rear = 0;

} else if (cq->rear == MAX - 1 && cq->front != 0) {

cq->rear = 0;

} else {

cq->rear++;

}

cq->items[cq->rear] = element;

printf("Element %c inserted successfully.\n", element);

}

// Function to delete an element from the Circular Queue

char deleteElement(struct CircularQueue \*cq) {

if (cq->front == -1) {

printf("Queue is Empty (Underflow)\n");

return '\0';

}

char deletedElement = cq->items[cq->front];

if (cq->front == cq->rear) {

cq->front = cq->rear = -1;

} else if (cq->front == MAX - 1) {

cq->front = 0;

} else {

cq->front++;

}

return deletedElement;

}

// Function to display the status of Circular Queue

void displayQueue(struct CircularQueue cq) {

if (isEmpty(cq)) {

printf("Queue is Empty.\n");

return;

}

printf("Circular Queue elements: ");

int i;

for (i = cq.front; i != cq.rear; i = (i + 1) % MAX) {

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

}

printf("%c\n", cq.items[i]);

}

// Function to check if Circular Queue is full

int isFull(struct CircularQueue cq) {

return (cq.front == 0 && cq.rear == MAX - 1) || (cq.rear == (cq.front - 1) % (MAX - 1));

}

// Function to check if Circular Queue is empty

int isEmpty(struct CircularQueue cq) {

return cq.front == -1;

}

Sample o/p

Circular Queue Operations:

1. Insert Element

2. Delete Element

3. Demonstrate Overflow and Underflow

4. Display Queue

5. Exit

Enter your choice: 1

Enter the element to be inserted: 10

Element 1 inserted successfully.

Circular Queue Operations:

1. Insert Element

2. Delete Element

3. Demonstrate Overflow and Underflow

4. Display Queue

5. Exit

Enter your choice: Invalid choice. Please enter a valid option.

Circular Queue Operations:

1. Insert Element

2. Delete Element

3. Demonstrate Overflow and Underflow

4. Display Queue

5. Exit

Enter your choice: 1

Enter the element to be inserted: 20

Element 2 inserted successfully.

Circular Queue Operations:

1. Insert Element

2. Delete Element

3. Demonstrate Overflow and Underflow

4. Display Queue

5. Exit

Enter your choice: Invalid choice. Please enter a valid option.

Circular Queue Operations:

1. Insert Element

2. Delete Element

3. Demonstrate Overflow and Underflow

4. Display Queue

5. Exit

Enter your choice: 1

Enter the element to be inserted: 30

Element 3 inserted successfully.

Circular Queue Operations:

1. Insert Element

2. Delete Element

3. Demonstrate Overflow and Underflow

4. Display Queue

5. Exit

Enter your choice: Invalid choice. Please enter a valid option.

Circular Queue Operations:

1. Insert Element

2. Delete Element

3. Demonstrate Overflow and Underflow

4. Display Queue

5. Exit

Enter your choice: 4

Circular Queue elements: 1 2 3

Circular Queue Operations:

1. Insert Element

2. Delete Element

3. Demonstrate Overflow and Underflow

4. Display Queue

5. Exit

Enter your choice: 1

Enter the element to be inserted: 40

Element 4 inserted successfully.

Circular Queue Operations:

1. Insert Element

2. Delete Element

3. Demonstrate Overflow and Underflow

4. Display Queue

5. Exit

Enter your choice: Invalid choice. Please enter a valid option.

Circular Queue Operations:

1. Insert Element

2. Delete Element

3. Demonstrate Overflow and Underflow

4. Display Queue

5. Exit

Enter your choice: 1

Enter the element to be inserted: 50

Element 5 inserted successfully.

Circular Queue Operations:

1. Insert Element

2. Delete Element

3. Demonstrate Overflow and Underflow

4. Display Queue

5. Exit

Enter your choice: Invalid choice. Please enter a valid option.

Circular Queue Operations:

1. Insert Element

2. Delete Element

3. Demonstrate Overflow and Underflow

4. Display Queue

5. Exit

Enter your choice: 1

Enter the element to be inserted: 60

Queue is Full (Overflow)

1. Design, Develop and Implement a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes.
2. Represent and Evaluate a Polynomial P(x,y,z) = 6x2y2z-4yz5+3x3yz+2xy5z2xyz3
3. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z)

Support the program with appropriate functions for each of the above operations.

#include <stdio.h>

#include <stdlib.h>

// Node structure for polynomial terms

struct Node {

int coefficient;

int exponent\_x;

int exponent\_y;

int exponent\_z;

struct Node\* next;

};

// Function prototypes

void insertTerm(struct Node\*\* head, int coeff, int exp\_x, int exp\_y, int exp\_z);

void displayPolynomial(struct Node\* head);

int evaluatePolynomial(struct Node\* head, int x, int y, int z);

struct Node\* addPolynomials(struct Node\* poly1, struct Node\* poly2);

int main() {

// Polynomial P(x, y, z) = 6x^2y^2z - 4yz^5 + 3x^3yz + 2xy^5z^2xyz^3

struct Node\* polyP = NULL;

insertTerm(&polyP, 6, 2, 2, 1);

insertTerm(&polyP, -4, 0, 1, 5);

insertTerm(&polyP, 3, 3, 1, 1);

insertTerm(&polyP, 2, 1, 5, 2);

printf("Polynomial P(x, y, z): ");

displayPolynomial(polyP);

// Polynomial POLY1(x, y, z) = 2x^2y^2z^3 + 3xyz - 5xy^2z

struct Node\* poly1 = NULL;

insertTerm(&poly1, 2, 2, 1, 3);

insertTerm(&poly1, 3, 1, 1, 1);

insertTerm(&poly1, -5, 1, 2, 1);

printf("\nPolynomial POLY1(x, y, z): ");

displayPolynomial(poly1);

// Polynomial POLY2(x, y, z) = -x^2y^2z + 4yz^5 + 2xy^5z^2

struct Node\* poly2 = NULL;

insertTerm(&poly2, -1, 2, 2, 1);

insertTerm(&poly2, 4, 0, 1, 5);

insertTerm(&poly2, 2, 1, 5, 2);

printf("\nPolynomial POLY2(x, y, z): ");

displayPolynomial(poly2);

// Evaluate Polynomial P(x, y, z) at x=1, y=2, z=3

int result = evaluatePolynomial(polyP, 1, 2, 3);

printf("\nResult of P(1, 2, 3): %d\n", result);

// Sum of POLY1 and POLY2

struct Node\* polySum = addPolynomials(poly1, poly2);

printf("\nSum of POLY1 and POLY2: ");

displayPolynomial(polySum);

// Clean up memory

free(polyP);

free(poly1);

free(poly2);

free(polySum);

return 0;

}

// Function to insert a term into the polynomial

void insertTerm(struct Node\*\* head, int coeff, int exp\_x, int exp\_y, int exp\_z) {

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

if (!newNode) {

printf("Memory allocation failed.\n");

exit(1);

}

newNode->coefficient = coeff;

newNode->exponent\_x = exp\_x;

newNode->exponent\_y = exp\_y;

newNode->exponent\_z = exp\_z;

newNode->next = \*head;

\*head = newNode;

}

// Function to display the polynomial

void displayPolynomial(struct Node\* head) {

if (!head) {

printf("Empty polynomial.\n");

return;

}

while (head->next != NULL) {

printf("%dx^%dy^%dz^%d + ", head->coefficient, head->exponent\_x, head->exponent\_y, head->exponent\_z);

head = head->next;

}

printf("%dx^%dy^%dz^%d\n", head->coefficient, head->exponent\_x, head->exponent\_y, head->exponent\_z);

}

// Function to evaluate the polynomial at given values of x, y, and z

int evaluatePolynomial(struct Node\* head, int x, int y, int z) {

int result = 0;

while (head != NULL) {

result += head->coefficient \* pow(x, head->exponent\_x) \* pow(y, head->exponent\_y) \* pow(z, head->exponent\_z);

head = head->next;

}

return result;

}

// Function to add two polynomials and return the result

struct Node\* addPolynomials(struct Node\* poly1, struct Node\* poly2) {

struct Node\* polySum = NULL;

while (poly1 != NULL || poly2 != NULL) {

int coeff1 = (poly1 != NULL) ? poly1->coefficient : 0;

int exp\_x1 = (poly1 != NULL) ? poly1->exponent\_x : 0;

int exp\_y1 = (poly1 != NULL) ? poly1->exponent\_y : 0;

int exp\_z1 = (poly1 != NULL) ? poly1->exponent\_z : 0;

int coeff2 = (poly2 != NULL) ? poly2->coefficient : 0;

int exp\_x2 = (poly2 != NULL) ? poly2->exponent\_x : 0;

int exp\_y2 = (poly2 != NULL) ? poly2->exponent\_y : 0;

int exp\_z2 = (poly2 != NULL) ? poly2->exponent\_z : 0;

int sumCoeff = coeff1 + coeff2;

int sumExp\_x = (exp\_x1 > exp\_x2) ? exp\_x1 : exp\_x2;

int sumExp\_y = (exp\_y1 > exp\_y2) ? exp\_y1 : exp\_y2;

int sumExp\_z = (exp\_z1 > exp\_z2) ? exp\_z1 : exp\_z2;

insertTerm(&polySum, sumCoeff, sumExp\_x, sumExp\_y, sumExp\_z);

if (poly1 != NULL) {

poly1 = poly1->next;

}

if (poly2 != NULL) {

poly2 = poly2->next;

}

}

return polySum;

}

Sample o/p

Polynomial P(x, y, z): 2x^1y^5z^2 + 3x^3y^1z^1 + -4x^0y^1z^5 + 6x^2y^2z^1

Polynomial POLY1(x, y, z): -5x^1y^2z^1 + 3x^1y^1z^1 + 2x^2y^1z^3

Polynomial POLY2(x, y, z): 2x^1y^5z^2 + 4x^0y^1z^5 + -1x^2y^2z^1

Result of P(1, 2, 3): -1278(2(1)(2)5(3)2+3(1)3(2)1(3)1−4(1)(2)1(3)5+6(1)2(2)2(3)1)

Sum of POLY1 and POLY2: 1x^2y^2z^3 + 7x^1y^1z^5 + -3x^1y^5z^2

1. Design and Develop a program in C for the following operations on Binary Search Tree (BST) of Integers.

a. Create a BST of N Integers

b. Traverse the BST using Inorder, Preorder and Post Order techniques

c. Search a KEY element in BST and display the appropriate message

#include <stdio.h>

#include <stdlib.h>

// Structure for a BST node

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

// Function to create a new node

struct Node\* createNode(int key) {

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

newNode->data = key;

newNode->left = newNode->right = NULL;

return newNode;

}

// Function to insert a key into BST

struct Node\* insert(struct Node\* root, int key) {

if (root == NULL) {

return createNode(key);

}

if (key < root->data) {

root->left = insert(root->left, key);

} else if (key > root->data) {

root->right = insert(root->right, key);

}

return root;

}

// Function to perform inorder traversal

void inorderTraversal(struct Node\* root) {

if (root != NULL) {

inorderTraversal(root->left);

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

inorderTraversal(root->right);

}

}

// Function to perform preorder traversal

void preorderTraversal(struct Node\* root) {

if (root != NULL) {

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

preorderTraversal(root->left);

preorderTraversal(root->right);

}

}

// Function to perform postorder traversal

void postorderTraversal(struct Node\* root) {

if (root != NULL) {

postorderTraversal(root->left);

postorderTraversal(root->right);

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

}

}

// Function to search for a key in BST

struct Node\* search(struct Node\* root, int key) {

if (root == NULL || root->data == key) {

return root;

}

if (key < root->data) {

return search(root->left, key);

} else {

return search(root->right, key);

}

}

// Main function

int main() {

struct Node\* root = NULL;

int N, key;

// Input: Create a BST of N Integers

printf("Enter the number of elements in the BST: ");

scanf("%d", &N);

printf("Enter the elements:\n");

for (int i = 0; i < N; i++) {

scanf("%d", &key);

root = insert(root, key);

}

// Output: Traverse the BST using Inorder, Preorder, and Postorder techniques

printf("\nInorder Traversal: ");

inorderTraversal(root);

printf("\nPreorder Traversal: ");

preorderTraversal(root);

printf("\nPostorder Traversal: ");

postorderTraversal(root);

// Input: Search a KEY element in BST

printf("\nEnter the element to search: ");

scanf("%d", &key);

// Output: Display the appropriate message

struct Node\* result = search(root, key);

if (result != NULL) {

printf("Element %d found in the BST.\n", key);

} else {

printf("Element %d not found in the BST.\n", key);

}

return 0;

}

Sample o/p

Enter the number of elements in the BST: 3

Enter the elements:

2

1

3

Inorder Traversal: 1 2 3

Preorder Traversal: 2 1 3

Postorder Traversal: 1 3 2

Enter the element to search: 2

Element 2 found in the BST.