DELHI TECHNOLOGICAL UNIVERSITY

# (FORMELY DELHI COLLEGE OF ENGINEERING)

**Department Of Software Engineering**



**DATA STRUCTURES**

**(SE-203)**

SUBMITTED TO: SUBMITTED BY:

Mr. Ankur Narwal Arjun Yadav 23/SE/32

Department Of Software Engineering

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Objective** | **Date** | **Sign** |
| **1.** | **To reverse an array of characters** | **21/08/24** |  |
| **2.** | **To perform various operations on array:**  **Insert element**  **Delete element**  **Find largest element**  **Find smallest element** | **04/09/24** |  |
| **3.** | **To perform string operations:**  **Merge 2 strings**  **Reverse a string**  **Find and replace substring** | **09/10/24** |  |
| **4.** | **Write a program to implement character stack using an array**   * **Push** * **Popup function using boundary condition** * **Also write parenthesis correctness in a string array** | **16/10/24** |  |
| **5.** | **Write a program to display, insert and delete element and remove duplicates to a circular queue using menu driven program. Also check for overflow and underflow condition.** | **16/10/24** |  |
| **6.** | **write a program for displaying, inserting and deleting element to doubly link list** | **16/10/24** |  |
| **7.** | **choose a unique expression and store it in a binary tree. Use appropriate tree traversal to generate postfix, prefix and infix** | **16/10/24** |  |

**EXPERIMENT-1**

**AIM:** To take input from user and reverse an array.

**CODE:**

#include<stdio.h>

int main(){

    int n, arr[n], i;

    printf("Enter the size of the array: ");

    scanf("%d", &n);

    printf("Enter the elements: ");

    for(i = 0; i < n; i++)

    {

        scanf("%d", &arr[i]);

    }

    int rev[n], j = 0;

    for(i = n-1; i >= 0; i--)

    {

        rev[j] = arr[i];

        j++;

    }

    printf("The Reversed array: ");

    for(i = 0; i < n; i++)

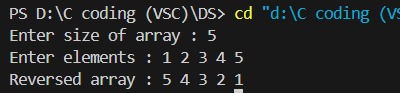
    {

        printf("%d ", rev[i]);

    }

}

**OUTPUT:**

****

**EXPERIMENT-2**

**AIM:** To perform various operations on array:

* Insert element
* Delete element
* Find largest element
* Find smallest element

**CODE:**

#include<stdio.h>

#define MAX\_size 100

int size = 0;

void inputarr(int arr[]);

void deletearr(int arr[]);

void largest(int arr[]);

void smallest(int arr[]);

void display(int arr[]);

void inputarr(int arr[]) {

    int n, elem;

    printf("Enter the number of elements you want to add: ");

    scanf("%d", &n);

    if (size + n > MAX\_size) {

        printf("Cannot add more elements. Array limit reached.\n");

        return;

    }

    for (int i = 0; i < n; i++) {

        printf("Enter element %d: ", size + 1);

        scanf("%d", &elem);

        arr[size++] = elem;

    }

}

void deletearr(int arr[]){

    if (size == 0) {

        printf("Array is empty. Nothing to delete.\n");

        return;

    }

    int num;

    printf("Enter the element to delete: ");

    scanf("%d", &num);

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

        if (arr[i]==num){

            arr[i] = arr[i + 1];}

    }

    size--;

    printf("Element deleted.\n");

}

void largest(int arr[]){

    int largest=-1e7;

    for(int i=0;i<size;i++){

        if(largest<arr[i]){

            largest=arr[i];

        }

    }

    printf("largest element is %d \n",largest);

}

void display(int arr[]){

    for(int i=0;i<size;i++){

        printf("%d",arr[i]);

    }

}

void smallest(int arr[]){

    int smallest=1e7;

    for(int i=0;i<size;i++){

        if(smallest>arr[i]){

            smallest=arr[i];

        }

    }

    printf("largest element is %d \n",smallest);

}

int main(){

    int arr[MAX\_size];

    int flag=1;

    while(flag){

        printf("Please Select an input \n");

        printf("1. Enter elements in a one dimensional array \n");

        printf("2. delete element in a one dimensional array (have all conditions, beginning, last, middle index)  \n");

        printf("3, Find the largest element \n");

        printf("4. Find the smallest element \n");

        printf("5. diaplay\n");

        printf("6.exit \n");

        int n;

        scanf("%d",&n);

        switch(n){

            case 1:

            inputarr(arr);

            break;

            case 2:

            deletearr(arr);

            break;

            case 3:

            largest(arr);

            break;

            case 4:

            smallest(arr);

            break;

            case 5:

            display(arr);

            case 6:

            flag=0;

            break;

            default:

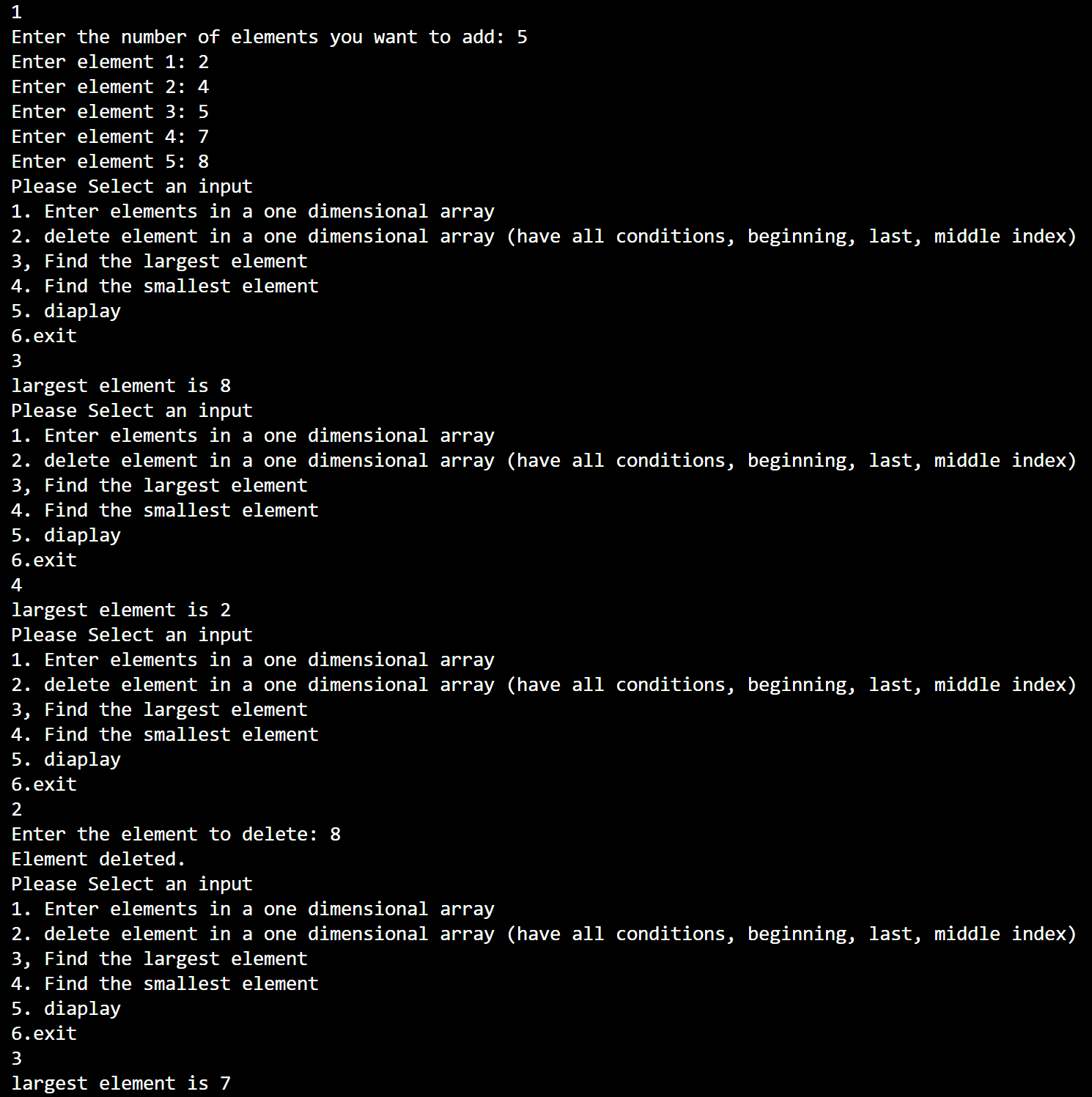
            printf("invalid input");

        }

    }

}

**OUTPUT:**

****

**EXPERIMENT-3**

**AIM:** To perform string operations:

* Merge 2 strings
* Reverse a string
* Find and replace substring

**CODE:**

#include <stdio.h>

#include <string.h>

void merge\_strings(char string1[], char string2[], int a, int b) {

    char merge[200];

    for (int i = 0; i < a; i++) {

        merge[i] = string1[i];

    }

    for (int j = 0; j < b; j++) {

        merge[j + a] = string2[j];

    }

    merge[a + b] = '\0';

    printf("Merged string is: %s\n", merge);

}

void substring\_replace(char string1[], char string2[], char string3[]) {

    char ans[1000] = {0};

    int ans\_idx = 0;

    int i = 0;

    while (i < strlen(string1)) {

        int found = 1;

        for (int j = 0; j < strlen(string2); j++) {

            if (string1[i + j] != string2[j]) {

                found = 0;

                break;

            }

        }

        if (found) {

            for (int k = 0; k < strlen(string3); k++) {

                ans[ans\_idx++] = string3[k];

            }

            i += strlen(string2);

        } else {

            ans[ans\_idx++] = string1[i];

            i++;

        }

    }

    ans[ans\_idx] = '\0';

    printf("Modified string: %s\n", ans);

}

void reverse\_string(char str[]) {

    int length = strlen(str);

    for (int i = 0; i < length / 2; i++) {

        char temp = str[i];

        str[i] = str[length - i - 1];

        str[length - i - 1] = temp;

    }

    printf("Reversed string: %s\n", str);

}

int main() {

    int choice;

    char string1[100], string2[100], string3[100];

    int flag = 1;

    while (flag) {

        printf("Choose an option:\n");

        printf("1. Merge two strings\n");

        printf("2. Reverse a string\n");

        printf("3. Replace a substring\n");

        printf("4. Exit\n");

        scanf("%d", &choice);

        getchar();

        switch (choice) {

            case 1:

                printf("Enter the first string: ");

                gets(string1);

                printf("Enter the second string: ");

                gets(string2);

                merge\_strings(string1, string2, strlen(string1), strlen(string2));

                break;

            case 2:

                printf("Enter the string to reverse: ");

                gets(string1);

                reverse\_string(string1);

                break;

            case 3:

                printf("Enter the original string: ");

                gets(string1);

                printf("Enter the substring to find: ");

                gets(string2);

                printf("Enter the replacement string: ");

                gets(string3);

                substring\_replace(string1, string2, string3);

                break;

            case 4:

                flag = 0;

                break;

            default:

                printf("Invalid choice.\n");

                break;

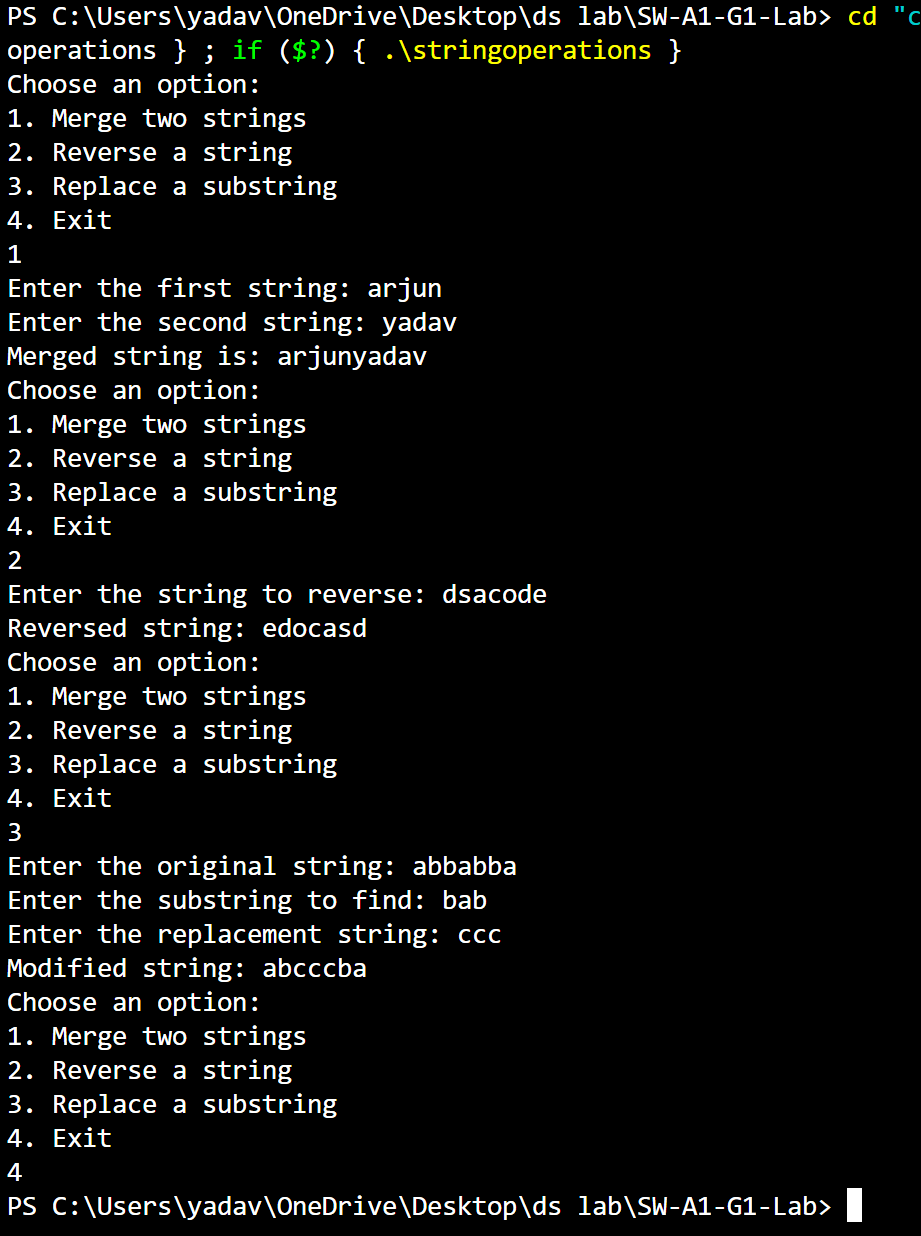
        }

    }

    return 0;

}

**OUTPUT:**

****

**EXPERIMENT 4**

**AIM:** Write a program to implement character stack using an array

* Push
* Popup function using boundary condition
* Also write parenthesis correctness in a string array

**CODE:**

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

#define MAX 100

typedef struct {

    char arr[MAX];

    int top;

} Stack;

void initialize(Stack \*stack) {

    stack->top = -1;

}

bool isEmpty(Stack \*stack) {

    return stack->top == -1;

}

void push(Stack \*stack, char x) {

    if (stack->top >= (MAX - 1)) {

        printf("Stack Overflow\n");

    } else {

        stack->arr[++(stack->top)] = x;

    }

}

char pop(Stack \*stack) {

    if (isEmpty(stack)) {

        printf("Stack Underflow\n");

        return '\0';

    } else {

        return stack->arr[(stack->top)--];

    }

}

bool areParenthesesBalanced(char str[]) {

    Stack stack;

    initialize(&stack);

    for (int i = 0; i < strlen(str); i++) {

        char ch = str[i];

        if (ch == '(' || ch == '{' || ch == '[') {

            push(&stack, ch);

        } else if (ch == ')' || ch == '}' || ch == ']') {

            if (isEmpty(&stack)) {

                return false;

            }

            char top = pop(&stack);

            if ((ch == ')' && top != '(') ||

                (ch == '}' && top != '{') ||

                (ch == ']' && top != '[')) {

                return false;

            }

        }

    }

    return isEmpty(&stack);

}

int main() {

    char expr[MAX];

    printf("Enter an expression: ");

    scanf("%s", expr);

    if (areParenthesesBalanced(expr)) {

        printf("The parentheses are balanced.\n");

    } else {

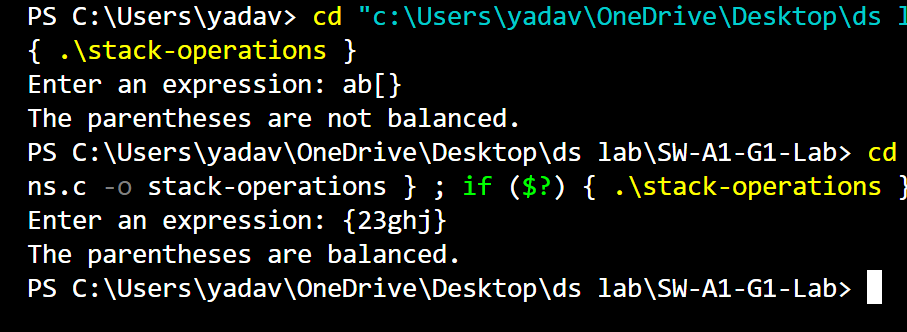
        printf("The parentheses are not balanced.\n");

    }

    return 0;

}

**OUTPUT:**



**EXPERIMENT 5**

**AIM:** Write a program to display, insert and delete element and remove duplicates to a circular queue using menu driven program. Also check for overflow and underflow condition.

**CODE:**

#include <stdio.h>

#include <stdlib.h>

// Arjujn yadav 16/10/24

#define MAX 100

int arr[MAX];

int front = -1;

int rear = -1;

void initialize() {

    front = -1;

    rear = -1;

}

int isFull() {

    return (front == (rear + 1) % MAX);

}

int isEmpty() {

    return (front == -1);

}

void display() {

    if (isEmpty()) {

        printf("Queue is empty.\n");

        return;

    }

    int i = front;

    printf("Queue elements: ");

    do {

        printf("%d ", arr[i]);

        i = (i + 1) % MAX;

    } while (i != (rear + 1) % MAX);

    printf("\n");

}

void insert(int n) {

    int a;

    for (int i = 0; i < n; i++) {

        if (isFull()) {

            printf("Queue Overflow. Cannot insert more elements.\n");

            return;

        }

        printf("Enter element %d: ", i + 1);

        scanf("%d", &a);

        if (isEmpty()) {

            front = rear = 0;

        } else {

            rear = (rear + 1) % MAX;

        }

        arr[rear] = a;

    }

}

void deleteElement() {

    if (isEmpty()) {

        printf("Queue Underflow\n");

        return;

    }

    printf("Deleted element: %d\n", arr[front]);

    if (front == rear) {

        front = rear = -1;

    } else {

        front = (front + 1) % MAX;

    }

}

void removeDuplicates() {

    if (isEmpty()) {

        printf("Queue is empty.\n");

        return;

    }

    int i = front;

    int j;

    do {

        j = (i + 1) % MAX;

        while (j != (rear + 1) % MAX) {

            if (arr[i] == arr[j]) {

                for (int k = j; k != rear; k = (k + 1) % MAX) {

                    arr[k] = arr[(k + 1) % MAX];

                }

                rear = (rear - 1 + MAX) % MAX;

            } else {

                j = (j + 1) % MAX;

            }

        }

        i = (i + 1) % MAX;

    } while (i != rear);

}

int main() {

    int n, a;

    do {

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

        printf("1. Display elements\n");

        printf("2. Insert an element\n");

        printf("3. Delete an element\n");

        printf("4. Remove duplicates\n");

        printf("5. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &n);

        switch (n) {

            case 1:

                display();

                break;

            case 2:

                printf("Enter the no of element you want to insert: ");

                scanf("%d", &a);

                insert(a);

                break;

            case 3:

                deleteElement();

                break;

            case 4:

                removeDuplicates();

                break;

            case 5:

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

                break;

            default:

                printf("Invalid choice. Please try again.\n");

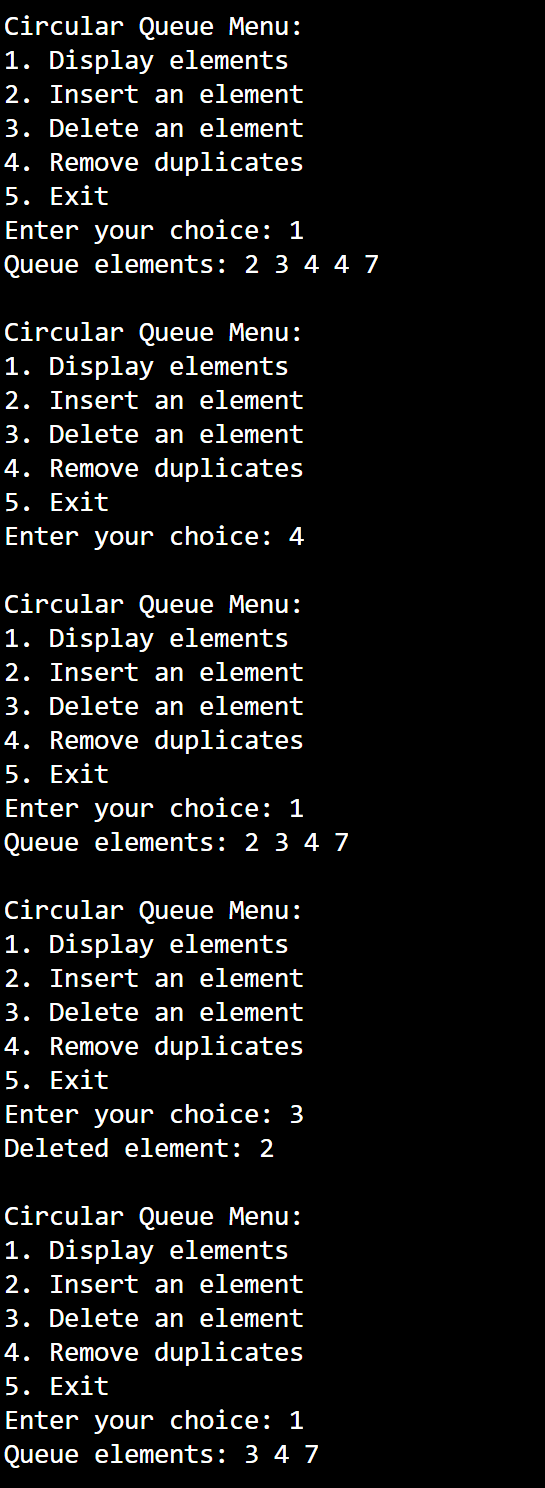
        }

    } while (n != 5);

    return 0;

}

**OUTPUT:**

****

**EXPERIMENT-6**

**AIM:** write a program for displaying, inserting and deleting element to doubly link list.

**CODE:**

#include <stdio.h>

#include <stdlib.h>

//arjun yadav 23/se/32

struct Node {

    int data;

    struct Node\* prev;

    struct Node\* next;

};

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;

}

void display(struct Node\* head) {

    struct Node\* temp = head;

    if (temp == NULL) {

        printf("List is empty.\n");

        return;

    }

    printf("Doubly Linked List: ");

    while (temp != NULL) {

        printf("%d ", temp->data);

        temp = temp->next;

    }

    printf("\n");

}

void insertdata(struct Node\*\* head, int data) {

    struct Node\* newNode = createNode(data);

    if (\*head == NULL) {

        \*head = newNode;

    } else {

        newNode->next = \*head;

        (\*head)->prev = newNode;

        \*head = newNode;

    }

    printf("Inserted %d at the beginning.\n", data);

}

void deletedata(struct Node\*\* head) {

    if (\*head == NULL) {

        printf("List is empty. Nothing to delete.\n");

        return;

    }

    struct Node\* temp = \*head;

    \*head = (\*head)->next;

    if (\*head != NULL) {

        (\*head)->prev = NULL;

    }

    printf("Deleted %d from the list.\n", temp->data);

    free(temp);

}

int main() {

    struct Node\* head = NULL;

    int choice, data;

    while (1) {

        printf("\nDoubly Linked List Operations:\n");

        printf("1. Display list\n");

        printf("2. Insert data\n");

        printf("3. Delete data\n");

        printf("4. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                display(head);

                break;

            case 2:

                printf("Enter data to insert: ");

                scanf("%d", &data);

                insertdata(&head, data);

                break;

            case 3:

                deletedata(&head);

                break;

            case 4:

                printf("Exit\n");

                exit(0);

            default:

                printf("Invalid choice! Please try again.\n");

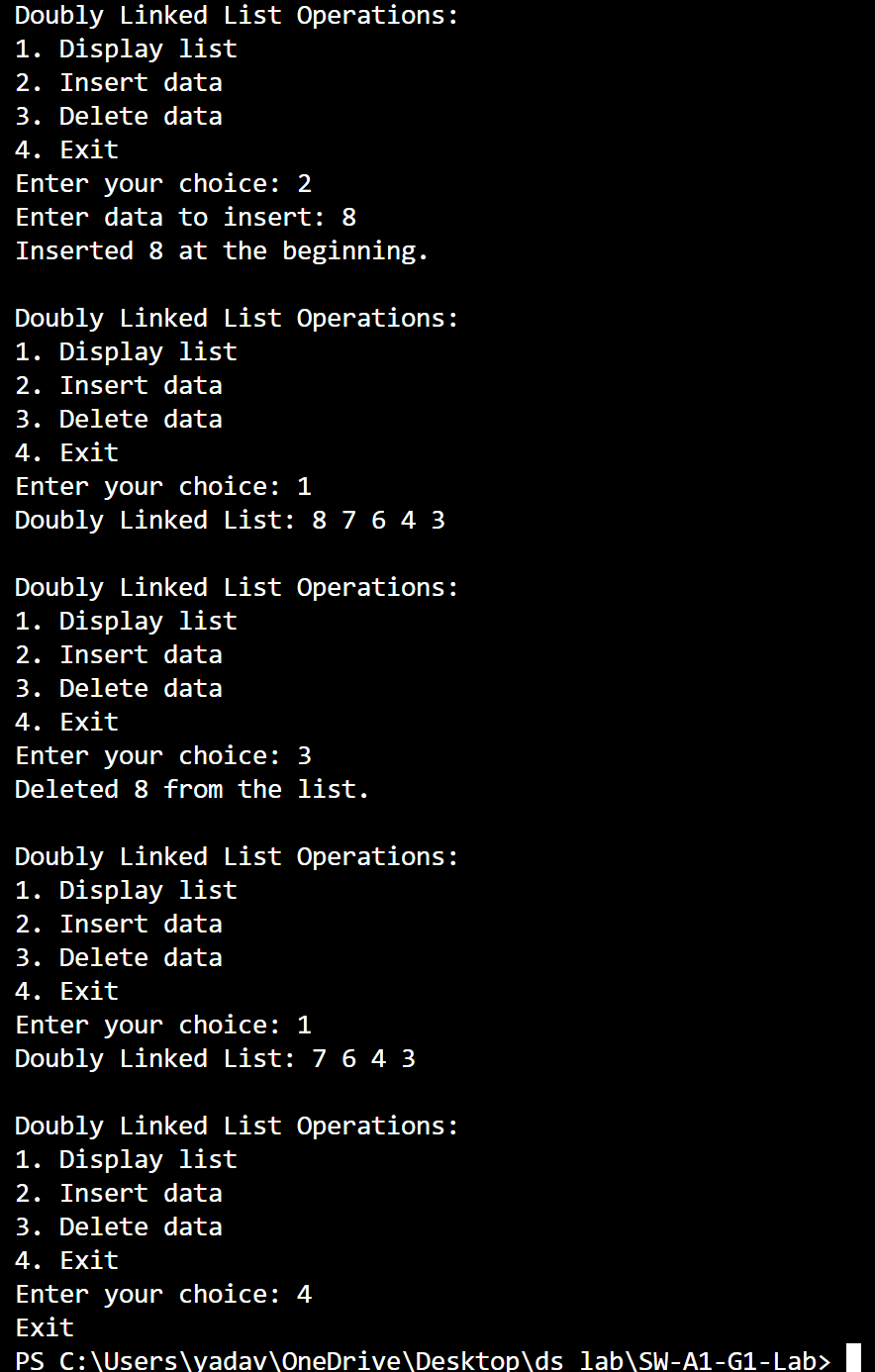
        }

    }

    return 0;

}

**OUTPUT:**

****

**EXPERIMENT-7**

**AIM:** Choose a unique expression and store it in a binary tree. Use appropriate tree traversal to generate postfix, prefix and infix.

**CODE:**

#include <stdio.h>

#include <stdlib.h>

//arjun yadav 23/se/32

struct Node {

    char value;

    struct Node\* left;

    struct Node\* right;

};

struct Node\* createNode(char value) {

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

    newNode->value = value;

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

    return newNode;

}

void preOrder(struct Node\* node) {

    if (node == NULL)

        return;

    printf("%c ", node->value);

    preOrder(node->left);

    preOrder(node->right);

}

void postOrder(struct Node\* node) {

    if (node == NULL)

        return;

    postOrder(node->left);

    postOrder(node->right);

    printf("%c ", node->value);

}

int main() {

    struct Node\* root = createNode('/');

    root->left = createNode('\*');

    root->right = createNode('D');

    root->left->left = createNode('A');

    root->left->right = createNode('+');

    root->left->right->left = createNode('B');

    root->left->right->right = createNode('C');

    int choice;

    while (1) {

        printf("\nDoubly Linked List Operations:\n");

        printf("1. prefix\n");

        printf("2. postfix\n");

        printf("3. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                printf("Prefix Expression: ");

                preOrder(root);

                printf("\n");

                break;

            case 2:

                printf("Postfix Expression: ");

                postOrder(root);

                printf("\n");

                break;

            case 3:

                printf("Exit\n");

                exit(0);

            default:

                printf("Invalid choice! Please try again.\n");

        }

    }

    return 0;

}

**OUTPUT:**

