DELHI TECHNOLOGICAL UNIVERSITY

# (FORMELY DELHI COLLEGE OF ENGINEERING)

**Department Of Software Engineering**



**DATA STRUCTURES**

**(SE-203)**

SUBMITTED TO: SUBMITTED BY:

Mr. Ankur Narwal Arnav Gupta 23/SE/34

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 reverse(int arr[], int n);

void printArr(int arr[], int n);

int main(){

    int n;

    printf("enter the length of array : ");

    scanf("%d",&n);

    int arr[n];

    for(int i=0;i<n;i++){

        int num;

        scanf("%d",&num);

        arr[i]=num;

    }

    reverse(arr,n);

    printArr(arr,n);

    return 0;

}

int reverse(int arr[], int n){

    for (int i=0; i<(n/2);i++){

        int v1=arr[i];

        int v2=arr[n-i-1];

        arr[i]=v2;

        arr[n-i-1]=v1;

    }

}

void printArr(int arr[], int n){

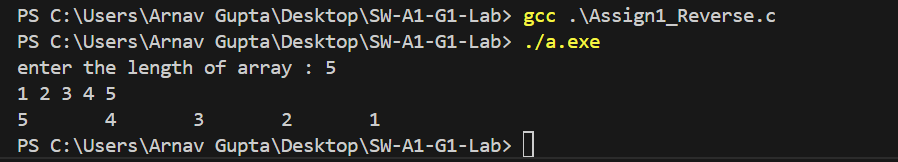
    for(int i=0 ;i<n; i++){

        printf("%d\t",arr[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>

void printNumbers(int arr[],int n){

    for(int i=0; i<n; i++){

        printf("%d\t",arr[i]);

    }

}

int main(){

    int n;

    printf("enter the length : \n");

    scanf("%d", &n);

    int arr[n];

    int size=0;

    int counter=0;

    printf("1. for adding element \n");

    printf("2. for deleting element \n");

    printf("3. largest element \n");

    printf("4. smallest element \n");

    printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \n");

    scanf("%d", &counter);

    do{

        if(counter==1){

            int num;

            printf("enter the number  : ");

            scanf("%d", &num);

            arr[size]=num;

            size++;

        } else if (counter==2){

            int idx;

            printf("enter the index : ");

            scanf("%d",&idx);

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

                if(i>=idx){

                    arr[i]=arr[i+1];

                }

            }

            size--;

        } else if (counter==3){

            int maxm=-99999;

            for(int i=0;i<size;i++){

                if (arr[i]>maxm){

                    maxm=arr[i];

                }

            }

            printf(" \nmaxm is : %d", maxm);

        } else {

            int  minm = 99999;

            for(int i=0;i<size;i++){

                if(arr[i]<minm){

                    minm=arr[i];

                }

            }

            printf("\nminimum is : %d", minm);

        }

        printf("\nenter the option : ");

        scanf("%d", &counter);

    } while(counter>0 && counter<5);

    for(int i=0;i<size;i++){

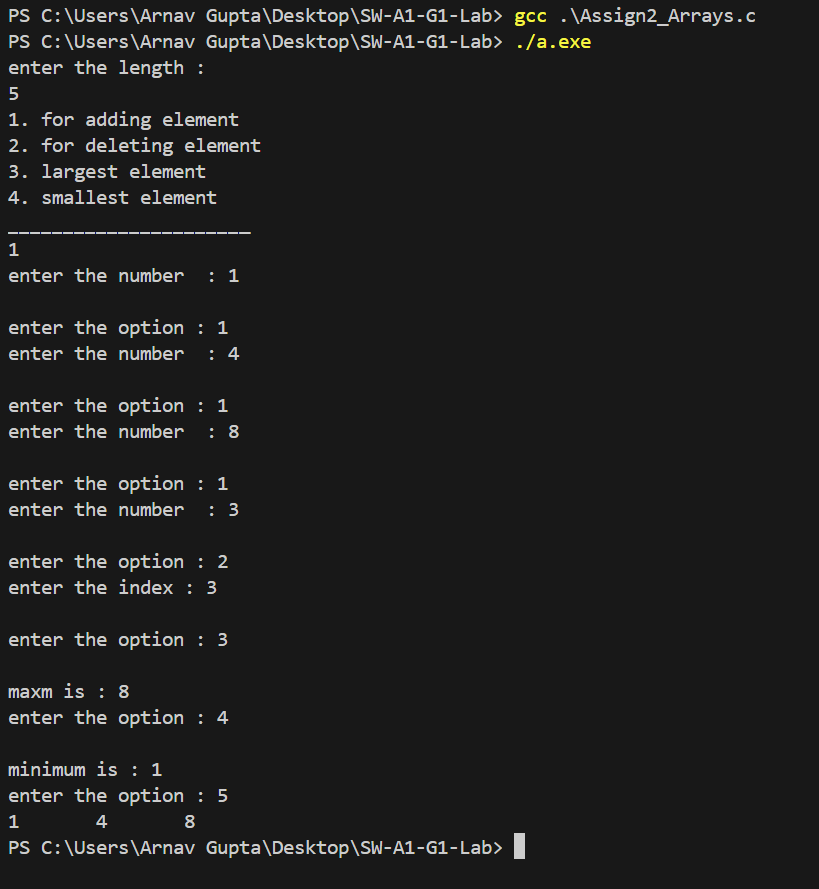
            printf("%d \t",arr[i]);

        }

    return 0;

}

**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>

#include<stdlib.h>

char\* merge(char\* s1, char \*s2){

    int i = strlen(s1);

    int j = 0;

    while (s2[j] != '\0') {

        s1[i] = s2[j];

        i++;

        j++;

    }

    s1[i] = '\0';

    return s1;

}

char\* reversal(char \* str){

    int i = strlen(str) - 1;

    int j = 0;

    char temp;

    while(j < i) {

        temp = str[j];

        str[j] = str[i];

        str[i] = temp;

        j++;

        i--;

    }

    return str;

}

char\* replace(char\* str, char\* sub, char \*new){

    int j = 0, index = 0, size = strlen(sub);

    int f1 = 0, f2 = 0;

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

        if(str[i] == sub[j]) {

            f1 = 1;

            int J = j;

            int k = i;

            while(size > 1) {

                if(str[k] != sub[J]) {

                    f2 = 1;

                    break;

                }

                J++;

                size--;

                k++;

            }

            index = i;

            break;

        }

    }

    if(f1 == 1 && f2 == 0) {

        int g = 0;

        while(new[g] != '\0') {

            str[index] = new[g];

            index++;

            g++;

        }

        return str;

    } else {

        return "No required substring found";

    }

}

int main() {

    int choice = 0;

    char s1[100];

    char s2[100];

    char s3[100];

    char s4[100];

    char sub[50];

    char new[50];

    printf("Options: \n");

    printf("1 : Merge \n");

    printf("2 : Reverse string \n");

    printf("3 : Find substring and replace with another \n");

    printf("Enter your choice: ");

    scanf("%d", &choice);

    getchar();

    do {

        if (choice == 1) {

            printf("Enter first string: ");

            fgets(s1, 100, stdin);

            s1[strcspn(s1, "\n")] = 0;

            printf("Enter second string: ");

            fgets(s2, 100, stdin);

            s2[strcspn(s2, "\n")] = 0;

            printf("Concatenated string is: %s\n", merge(s1, s2));

        } else if (choice == 2) {

            printf("Enter string: ");

            fgets(s3, 100, stdin);

            s3[strcspn(s3, "\n")] = 0;

            printf("The reversed string is: %s\n", reversal(s3));

        } else if (choice == 3) {

            printf("Enter string: ");

            fgets(s4, 100, stdin);

            s4[strcspn(s4, "\n")] = 0;

            printf("Enter substring to be found: ");

            fgets(sub, 50, stdin);

            sub[strcspn(sub, "\n")] = 0;

            printf("Enter substring to replace with: ");

            fgets(new, 50, stdin);

            new[strcspn(new, "\n")] = 0;

            printf("New string is: %s\n", replace(s4, sub, new));

        } else {

            printf("Wrong option\n");

        }

        printf("Enter your choice: ");

        scanf("%d", &choice);

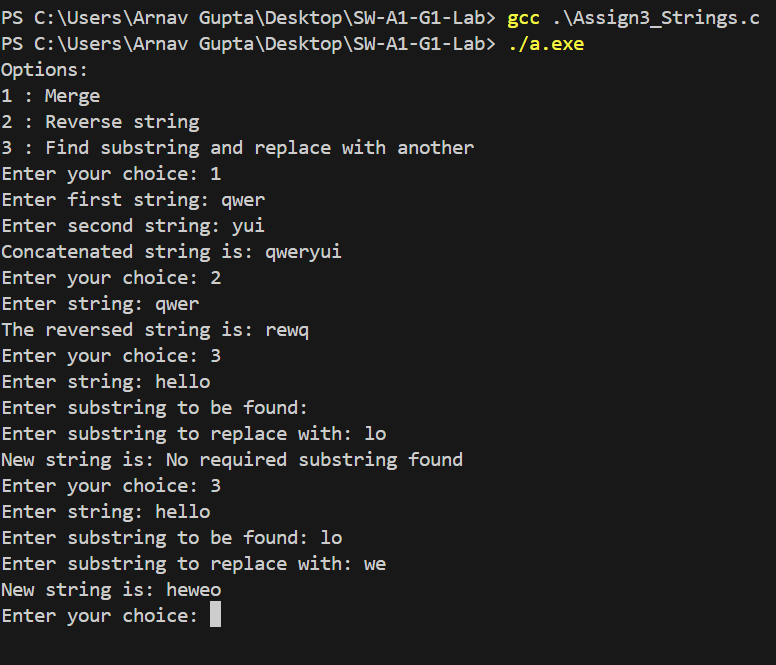
        getchar();

    } while (choice > 0 && choice < 4);

    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>

#define MAX\_SIZE 10

char data[MAX\_SIZE];

int top = -1;

int isEmpty() {

    return (top == -1);

}

int isFull() {

    return (top == MAX\_SIZE - 1);

}

void push(char c) {

    if (isFull()) {

        printf("Stack is full. Cannot push %c\n", c);

    } else {

        data[++top] = c;

    }

}

// Function to pop an element from the stack

char pop() {

    if (isEmpty()) {

        printf("Stack is empty. Cannot pop\n");

        return '\0';

    } else {

        return data[top--];

    }

}

int checkParenthesis(char str[]) {

    int i, len = 0;

    while (str[len] != '\0') {

        len++;

    }

    for (i = 0; i < len; i++) {

        if (str[i] == '(' || str[i] == '{' || str[i] == '[') {

            push(str[i]);

        } else if (str[i] == ')' || str[i] == '}' || str[i] == ']') {

            if (isEmpty()) {

                return 0;

            }

            char top = pop();

        }

    }

    if (isEmpty()) {

        return 1;

    } else {

        return 0;

    }

}

int main() {

    push('A');

    push('B');

    push('C');

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

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

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

    char str[] = "(A\*B+(c\*(D+E)))";

    char str1[] = "(a+b+(c-d\*(e+f))";

    if (checkParenthesis(str)) {

        printf("Parenthesis are balanced in the string : %s\n", str);

    } else {

        printf("Parenthesis are not balanced in the string : %s\n", str1);

    }

    if (checkParenthesis(str1)) {

        printf("Parenthesis are balanced in the string : %s\n", str);

    } else {

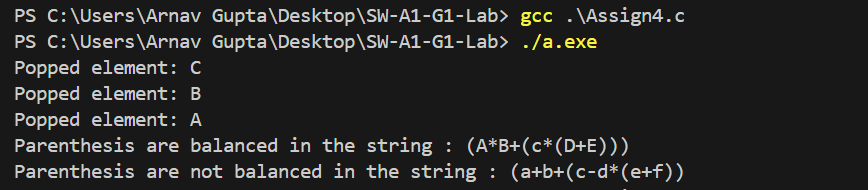
        printf("Parenthesis are not balanced in the string : %s\n", str1);

    }

    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>

#define MAX\_SIZE 5

int queue[MAX\_SIZE];

int front = 0, rear = 0;

void enqueue(int x) {

    if ((rear + 1) % MAX\_SIZE == front) {

        printf("Queue is full. Cannot insert %d\n", x);

    } else {

        queue[rear] = x;

        rear = (rear + 1) % MAX\_SIZE;

    }

}

int dequeue() {

    if (front == rear) {

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

        return -1;

    } else {

        int x = queue[front];

        front = (front + 1) % MAX\_SIZE;

        return x;

    }

}

void display() {

    int i = front;

    while (i != rear) {

        printf("%d ", queue[i]);

        i = (i + 1) % MAX\_SIZE;

    }

    printf("\n");

}

void removeDuplicates() {

    int i, j;

    for (i = front; i != rear; i = (i + 1) % MAX\_SIZE) {

        for (j = i + 1; j != rear; j = (j + 1) % MAX\_SIZE) {

            if (queue[i] == queue[j]) {

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

                    queue[k] = queue[(k + 1) % MAX\_SIZE];

                }

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

                j = (j - 1 + MAX\_SIZE) % MAX\_SIZE;

            }

        }

    }

}

int main() {

    int choice, x;

    while (1) {

        printf("1. Insert\n2. Delete\n3. Display\n4. Remove duplicates\n5. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

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

                scanf("%d", &x);

                enqueue(x);

                break;

            case 2:

                x = dequeue();

                if (x != -1) {

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

                }

                break;

            case 3:

                display();

                break;

            case 4:

                removeDuplicates();

                break;

            case 5:

                return 0;

            default:

                printf("Invalid choice\n");

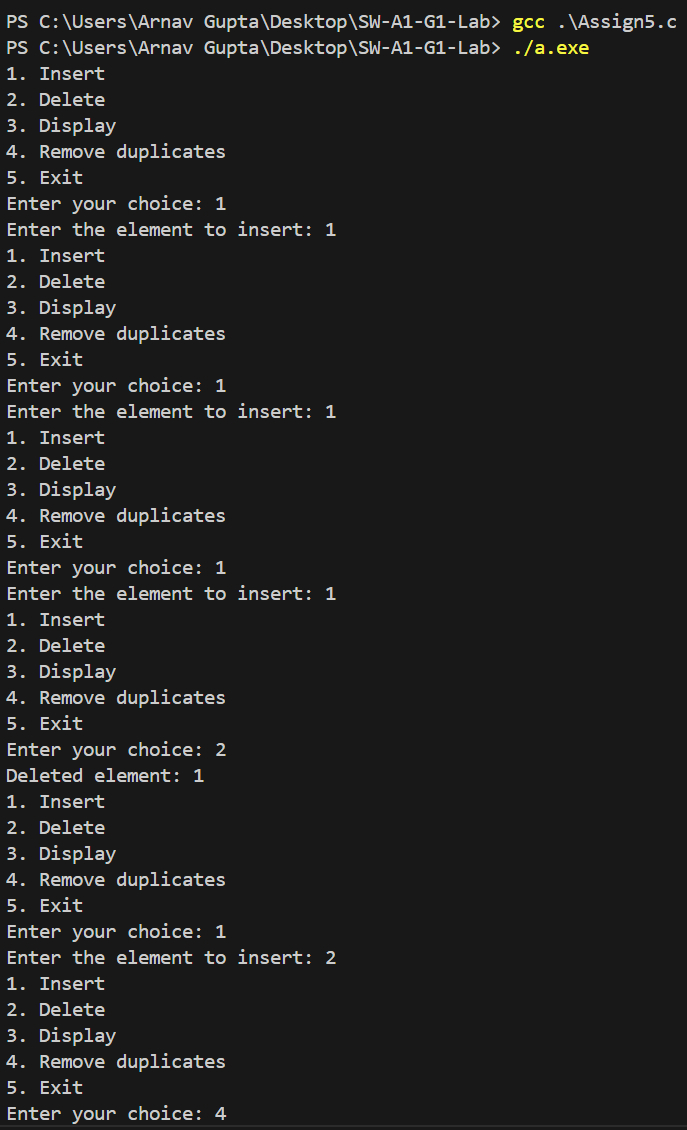
        }

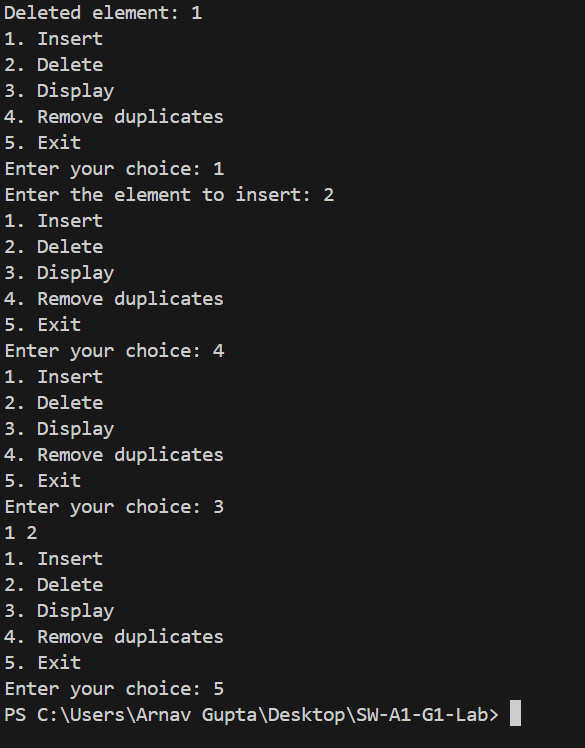
    }

    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>

typedef struct Node {

    int data;

    struct Node\* next;

    struct Node\* prev;

} Node;

Node\* head = NULL;

Node\* createNode(int data) {

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

    newNode->data = data;

    newNode->next = NULL;

    newNode->prev = NULL;

    return newNode;

}

void insertAtBeginning(int data) {

    Node\* newNode = createNode(data);

    if (head != NULL) {

        newNode->next = head;

        head->prev = newNode;

    }

    head = newNode;

}

void insertAtEnd(int data) {

    Node\* newNode = createNode(data);

    if (head == NULL) {

        head = newNode;

        return;

    }

    Node\* temp = head;

    while (temp->next != NULL) {

        temp = temp->next;

    }

    temp->next = newNode;

    newNode->prev = temp;

}

void deleteNode(int data) {

    Node\* temp = head;

    while (temp != NULL) {

        if (temp->data == data) {

            if (temp->prev != NULL) {

                temp->prev->next = temp->next;

            } else {

                head = temp->next;

            }

            if (temp->next != NULL) {

                temp->next->prev = temp->prev;

            }

            free(temp);

            return;

        }

        temp = temp->next;

    }

    printf("Node not found\n");

}

void displayList() {

    Node\* temp = head;

    while (temp != NULL) {

        printf("%d ", temp->data);

        temp = temp->next;

    }

    printf("\n");

}

int main() {

    int choice, data;

    while (1) {

        printf("1. Insert at beginning\n2. Insert at end\n3. Delete node\n4. Display list\n5. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                printf("Enter the data to insert: ");

                scanf("%d", &data);

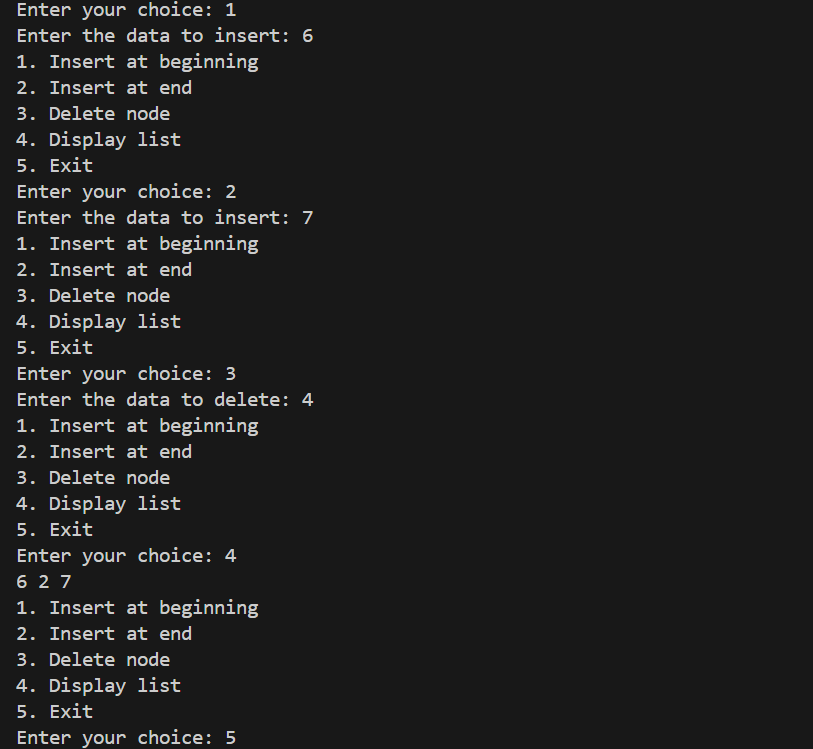
                insertAtBeginning(data);

                break;

            case 2:

                printf("Enter the data to insert: ");

                scanf("%d", &data);

****                insertAtEnd(data);

                break;

            case 3:

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

                scanf("%d", &data);

                deleteNode(data);

                break;

            case 4:

                displayList();

                break;

            case 5:

                return 0;

            default:

                printf("Invalid choice\n");

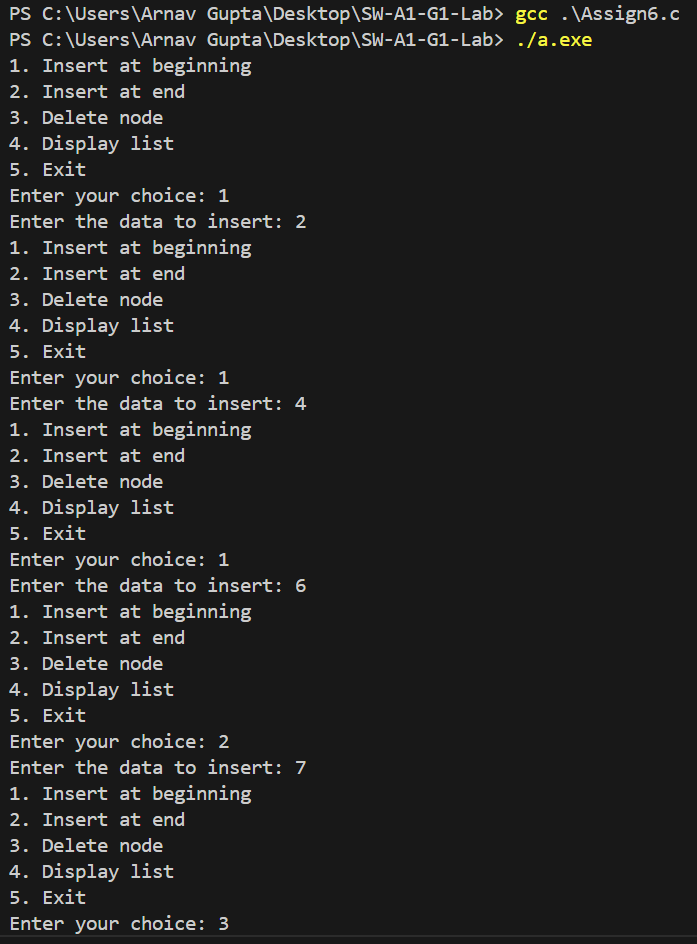
        }

    }

    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>

#include <string.h>

typedef struct Node {

    char data;

    struct Node\* left;

    struct Node\* right;

} Node;

Node\* createNode(char data) {

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

    newNode->data = data;

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

    return newNode;

}

int isOperator(char ch) {

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

}

Node\* constructTree(char postfix[]) {

    Node\* stack[100];

    int top = -1;

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

        if (!isOperator(postfix[i])) {

            stack[++top] = createNode(postfix[i]);

        } else {

            Node\* operatorNode = createNode(postfix[i]);

            operatorNode->right = stack[top--];

            operatorNode->left = stack[top--];

            stack[++top] = operatorNode;

        }

    }

    return stack[top];

}

void inorderTraversal(Node\* root) {

    if (root) {

        inorderTraversal(root->left);

        printf("%c ", root->data);

        inorderTraversal(root->right);

    }

}

void preorderTraversal(Node\* root) {

    if (root) {

        printf("%c ", root->data);

        preorderTraversal(root->left);

        preorderTraversal(root->right);

    }

}

void postorderTraversal(Node\* root) {

    if (root) {

        postorderTraversal(root->left);

        postorderTraversal(root->right);

        printf("%c ", root->data);

    }

}

int main() {

    char postfix[] = "ac\*w/wk\*+q-";

    Node\* root = constructTree(postfix);

    printf("Infix notation: ");

    inorderTraversal(root);

    printf("\n");

    printf("Prefix notation: ");

    preorderTraversal(root);

    printf("\n");

    printf("Postfix notation: ");

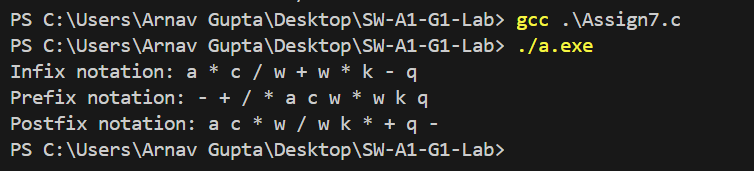
    postorderTraversal(root);

    printf("\n");

    return 0;

}

**OUTPUT:**

****