Practical -1

1. Write a program to calculate area of a Circle (A = πr2) (A)

#include<stdio.h>

float findArea(float r){

    return  3.14\*r\*r;

}

void main(){

    float radius=0;

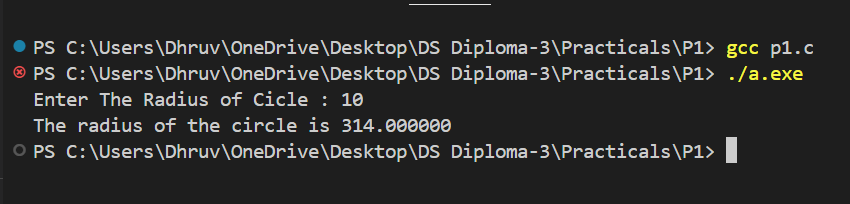
    printf("Enter The Radius of Cicle : ");

    scanf("%f",&radius);

    printf("The radius of the circle is %f",findArea(radius));

}

Output :-



2. Write a program to ƒind whether a number is odd or even(A).

#include<stdio.h>

void oddOrEven(int n){

    if(n%2==0)

        printf("Number is even");

    else

        printf("Number is odd");

}

void main(){

    int n;

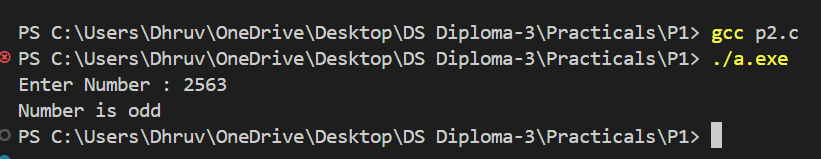
    printf("Enter Number : ");

    scanf("%d",&n);

    oddOrEven(n);

}

Output :-



3. Write a program to ƒind factorial of a number. (Using Loop) (A).

#include<stdio.h>

int fact(int n){

    int fact=1,i;

    for(i=1;i<=n;i++){

        fact\*=i;

    }

    return fact;

}

void main(){

    int n;

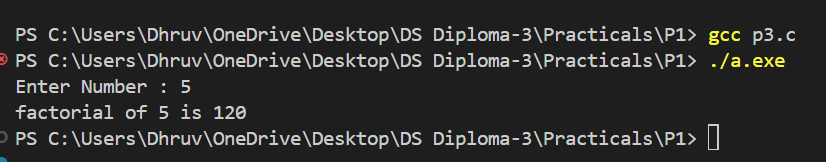
    printf("Enter Number : ");

    scanf("%d",&n);

    printf("factorial of %d is %d",n,fact(n));

}

Output:-



4. Write a program to ƒind power of a number using loop.(A)

#include <stdio.h>

int power(int base, int exponent) {

    int result = 1;

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

        result \*= base;

    }

    return result;

}

int main() {

    int base, exponent;

    // Input from user

    printf("Enter the base: ");

    scanf("%d", &base);

    printf("Enter the exponent: ");

    scanf("%d", &exponent);

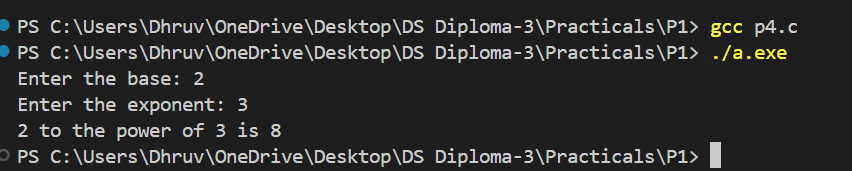
    int result = power(base, exponent);

    printf("%d to the power of %d is %d\n", base, exponent, result);

    return 0;

}

Output:-



5. Write a program to ƒind factors of a given number.(B)

#include<stdio.h>

void factors(int n){

    int i;

    for(i=1;i<=n;i++){

        if(n%i==0)

            printf("%d,",i);

    }

}

void main(){

    int n;

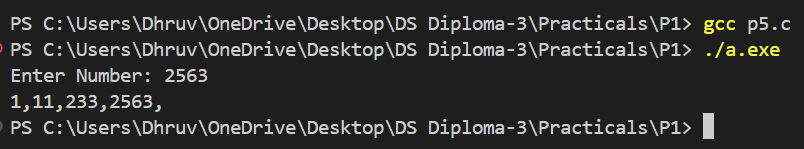
    printf("Enter Number: ");

    scanf("%d", &n);

    factors(n);

}

Output:-



6. Write a program to check whether a number is prime or not.(B)

#include<stdio.h>

void isPrime(int n){

    int i;

    int flag=0;

    for(i=2;i<n;i++){

        if(n%i==0){

            flag=1;

            break;

        }

    }

    if(flag==0)

        printf("number is prime  ");

    else

        printf("number is not prime");

}

void main(){

    int n;

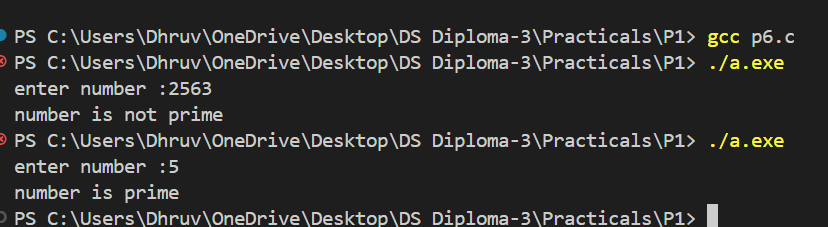
    printf("enter number");

    scanf("%d",&n);

    isPrime(n);

}

Output:-



7. Write a program to ƒind the sum of 1 + (1+2) + (1+2+3) + …+(1+2+3+4+….+n).(C)

#include<stdio.h>

int sum(int n){

    int i,j,sum=0;

    for(i=1;i<=n;i++){

        for(j=1;j<=i;j++){

            sum+=j;

        }

    }

    return sum;

}

void main(){

    int n;

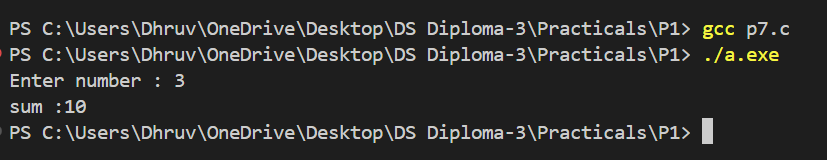
    printf("Enter number : ");

    scanf("%d", &n);

    printf("sum : %d",sum(n));

}

Output:-



8. Write a program to print Armstrong number from 1 to 1000. (C)

#include<stdio.h>

#include<stdbool.h>

int pow(int base,int exponent){

    int result=1,i;

    for(i=1;i<=exponent;i++){

        result\*=base;

    }

    return result;

}

bool armstrong(int n){

    int temp=n,len=0,sum=0;

    while(0<n){

        len++;

        n=(int)n/10;

    }

    n=temp;

    while(0<n){

        int digit = n%10;

        sum=sum+pow(digit,len);

        n=(int)n/10;

    }

    if(temp==sum) {

        return true;

}

    else{

        return false;

}

}

void main(){

    int i;

    for(i=1;i<1001;i++){

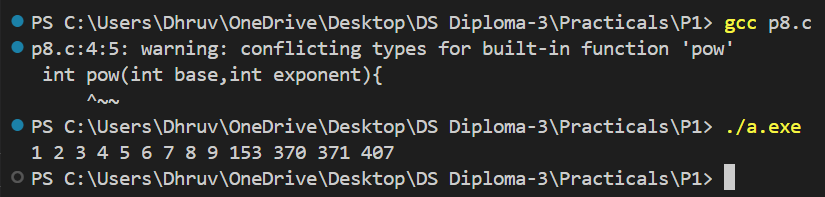
        if(armstrong(i))

            printf("%d ",i);

    }

}

Output:-



Practical -2

1. Write a program to read and display n numbers using an array. (A)

#include <stdio.h>

int main() {

    int n;

    // Input number of elements

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

    scanf("%d", &n);

    int arr[n];

    // Read elements into array

    printf("Enter %d numbers: ", n);

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    // Display elements of the array

    printf("The numbers are: ");

    for (int i = 0; i < n; i++) {

        printf("%d ", arr[i]);

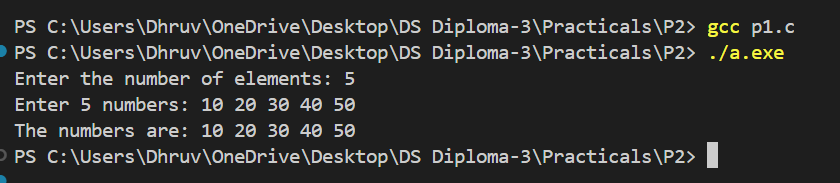
    }

    printf("\n");

    return 0;

}

Output :-



2. Write a program to calculate the sum of numbers from 1 to n. (A)

#include <stdio.h>

int main() {

    int n, sum = 0;

    // Input the value of n

    printf("Enter a number: ");

    scanf("%d", &n);

    // Calculate the sum of numbers from 1 to n

    for (int i = 1; i <= n; i++) {

        sum += i;

    }

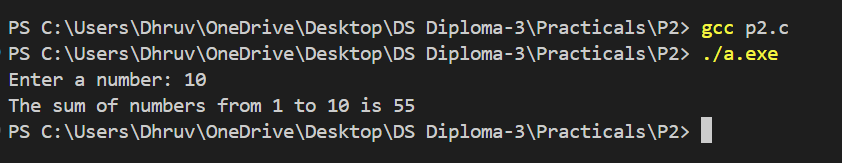
    // Display the sum

    printf("The sum of numbers from 1 to %d is %d\n", n, sum);

    return 0;

}

Output :-



3. Write a program to insert an element into an array at a given position.(A)

#include <stdio.h>

int main() {

    int n, pos, element;

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

    scanf("%d", &n);

    int arr[n+1]; // Extra space for the new element

    printf("Enter %d numbers: ", n);

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    // Input position and element to insert

    printf("Enter the position to insert (0 to %d): ", n-1);

    scanf("%d", &pos);

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

    scanf("%d", &element);

    // Insert element at the specified position

    for (int i = n; i > pos; i--) {

        arr[i] = arr[i-1];

    }

    arr[pos] = element;

    n++;

    printf("Array after insertion: ");

    for (int i = 0; i < n; i++) {

        printf("%d ", arr[i]);

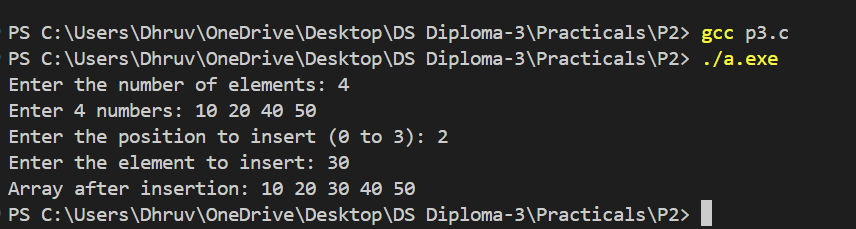
    }

    printf("\n");

    return 0;

}

Output :-



4. Write a program to delete a given element from an array. (A)

#include <stdio.h>

int main() {

    int n, element, pos = -1;

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

    scanf("%d", &n);

    int arr[n];

    printf("Enter %d numbers: ", n);

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

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

    scanf("%d", &element);

    // Find the position of the element to delete

    for (int i = 0; i < n; i++) {

        if (arr[i] == element) {

            pos = i;

            break;

        }

    }

    if (pos == -1) {

        printf("Element not found\n");

    } else {

        // Shift elements to the left to delete the element

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

            arr[i] = arr[i+1];

        }

        n--;

        printf("Array after deletion: ");

        for (int i = 0; i < n; i++) {

            printf("%d ", arr[i]);

        }

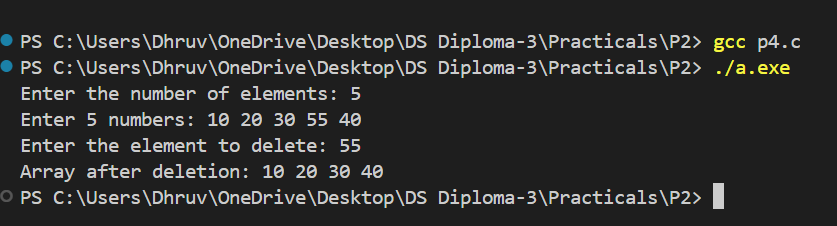
        printf("\n");

    }

    return 0;

}

Output :-



5. Write a program to ƒind whether the array contains a duplicate number or not. (B)

#include <stdio.h>

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

    for (int i = 0; i < n; i++) {

        for (int j = i + 1; j < n; j++) {

            if (arr[i] == arr[j]) {

                return 1; // Duplicate found

            }

        }

    }

    return 0; // No duplicate found

}

int main() {

    int n;

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

    scanf("%d", &n);

    int arr[n];

    printf("Enter %d numbers: ", n);

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    // Check for duplicates

    if (containsDuplicate(arr, n)) {

        printf("The array contains duplicate numbers.\n");

    } else {

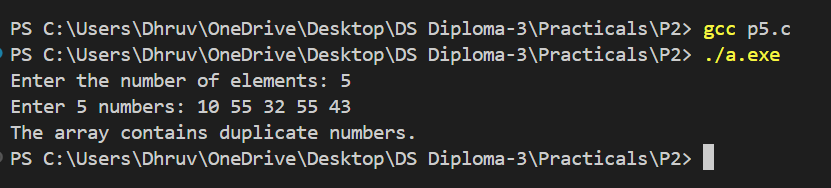
        printf("The array does not contain duplicate numbers.\n");

    }

    return 0;

}

Output :-



6. Write a program to delete a number from an array that is already sorted in an ascending order. (C)

#include <stdio.h>

int main() {

    int n, element, pos = -1;

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

    scanf("%d", &n);

    int arr[n];

    printf("Enter %d numbers (sorted in ascending order): ", n);

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

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

    scanf("%d", &element);

    // Find the position of the element to delete

    for (int i = 0; i < n; i++) {

        if (arr[i] == element) {

            pos = i;

            break;

        }

    }

    if (pos == -1) {

        printf("Element not found\n");

    } else {

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

            arr[i] = arr[i + 1];

        }

        n--;

        printf("Array after deletion: ");

        for (int i = 0; i < n; i++) {

            printf("%d ", arr[i]);

        }

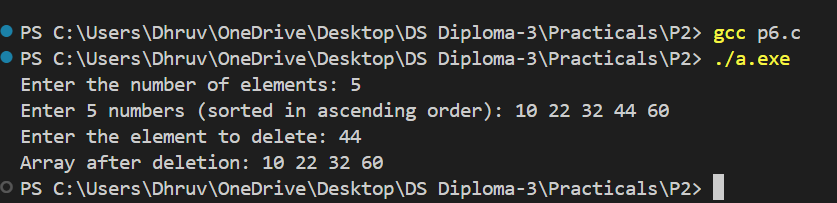
        printf("\n");

    }

    return 0;

}

Output :-



7. Write a program to delete duplicate numbers from an array. (C)

#include <stdio.h>

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

    if (n == 0 || n == 1) {

        return n;

    }

    int temp[n];

    int j = 0;

    for (int i = 0; i < n; i++) {

        int found = 0;

        for (int k = 0; k < j; k++) {

            if (arr[i] == temp[k]) {

                found = 1;

                break;

            }

        }

        if (!found) {

            temp[j++] = arr[i];

        }

    }

    for (int i = 0; i < j; i++) {

        arr[i] = temp[i];

    }

    return j;

}

int main() {

    int n;

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

    scanf("%d", &n);

    int arr[n];

    printf("Enter %d number : ", n);

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    printf("Original array: ");

    for (int i = 0; i < n; i++) {

        printf("%d ", arr[i]);

    }

    printf("\n");

    n = removeDuplicates(arr, n);

    printf("Array after removing duplicates: ");

    for (int i = 0; i < n; i++) {

        printf("%d ", arr[i]);

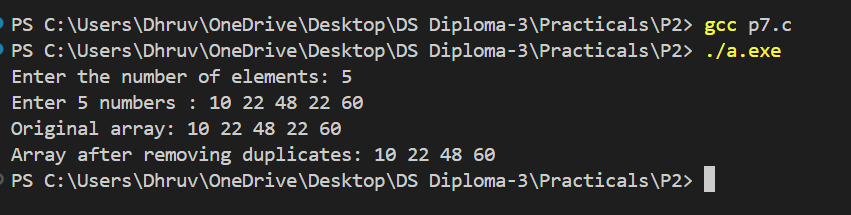
    }

    printf("\n");

    return 0;

}

Output :-



Practical -3

1.Write a program to ƒind the length of the given string. (A)

#include<stdio.h>

int countLength(char \*name){

    int len=0;

    while (\*name!='\0')

    {

        len=len+1;

        name++;

    }

    return len;

}

int main()

{

    char name[50];

    printf("Enter String :");

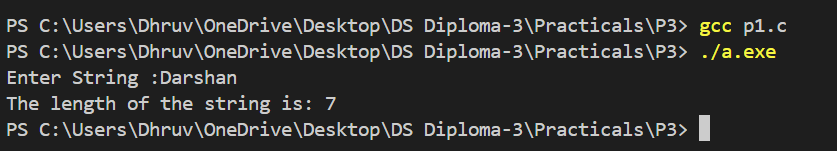
    gets(name);

    printf("The length of the string is: %d\n", countLength(name));

    return 0;

}

Output :-



2. Write a program to Copy one string to another string. (A)

#include <stdio.h>

void stringCopy(char \*str2, char \*str1) {

    while (\*str1 != '\0') {

        \*str2 = \*str1;

        str2++;

        str1++;

    }

    \*str2 = '\0';

}

int main() {

    char str1[100], str2[100];

    printf("Enter the source string: ");

    gets(str1);

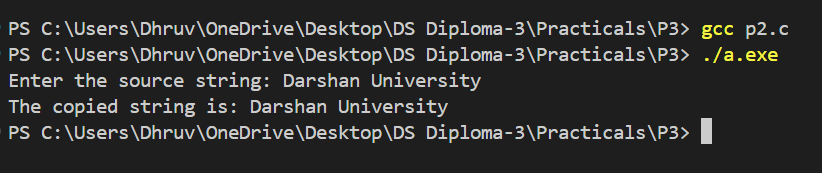
    stringCopy(str2, str1);

    printf("The copied string is: %s\n", str2);

    return 0;

}

Output :-



3. Write a program to perform the concatenation of two given strings. (A)

#include <stdio.h>

void stringConcatenate(char \*s1, char \*s2,char \*s3) {

    while (\*s1 != '\0') {

        \*s3=\*s1;

        s1++;

        s3++;

    }

    while (\*s2 != '\0') {

        \*s3 = \*s2;

        s3++;

        s2++;

    }

    \*s3 = '\0';

}

int main() {

    char str1[100], str2[100],str3[200];

    printf("Enter the first string: ");

    gets(str1);

    printf("Enter the second string: ");

    gets(str2);

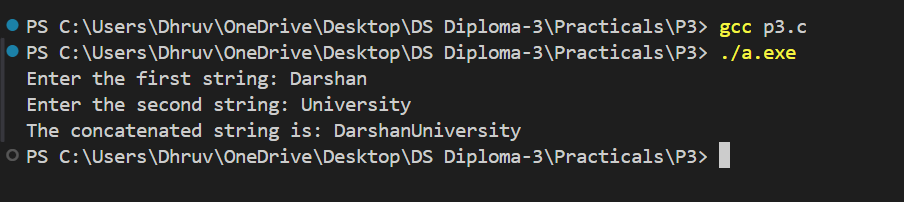
    stringConcatenate(str1, str2,str3);

    printf("The concatenated string is: %s\n", str3);

    return 0;

}

Output :-



4. Write a program to append a second string at the end of the ƒirst string. (A)

#include <stdio.h>

void stringAppend(char \*s1, char \*s2) {

    while (\*s1 != '\0') {

        s1++;

    }

    while (\*s2 != '\0') {

        \*s1 = \*s2;

        s1++;

        s2++;

    }

    \*s1 = '\0';

}

int main() {

    char str1[100], str2[100];

    printf("Enter the first string: ");

    gets(str1);

    printf("Enter the second string: ");

    gets(str2);

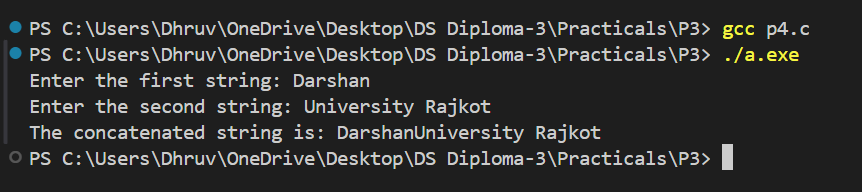
    stringAppend(str1, str2);

    printf("The concatenated string is: %s\n", str1);

    return 0;

}

Output :-



5. Write a program to insert a given substring into an existing string. (B)

#include<stdio.h>

void STRINSERTION(char \*s1,int p,char \*sub,char \*temp){

    while(p != 0)

    {

        \*temp++ = \*s1++;

        p--;

    }

    while(\*sub != '\0')

         \*temp++ = \*sub++;

    while(\*s1 != '\0')

    {

         \*temp++ = \*s1++;

    }

}

void main(){

    char new\_str[50];

    char s1[50] = "Hi Welcome";

    char sub[10] = "Hello";

    int pos = 3;

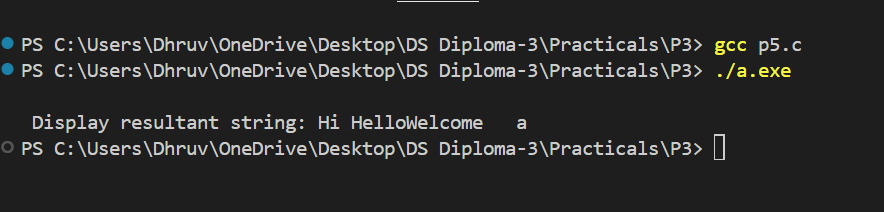
    STRINSERTION(s1, pos, sub, new\_str);

    printf("\n Display resultant string: ");

    puts(new\_str);

}

Output :-



6. Write a program to retrieve the substring from the given string. (B)

#include <stdio.h>

void substring(char str[], int pos, int len) {

    char substr[50];

    int i=0;

    pos=pos-1;

    while(len!=0){

        substr[i]=str[pos];

        i++;

        pos++;

        len--;

    }

    substr[i]='\0';

    printf("The retrieved substring is: %s\n", substr);

}

int main() {

    char str[100], substr[100];

    int pos, len;

    printf("Enter the main string: ");

    gets(str);

    printf("Enter the starting position of the substring: ");

    scanf("%d", &pos);

    printf("Enter the length of the substring: ");

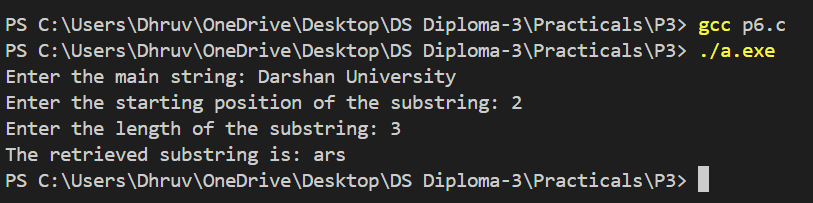
    scanf("%d", &len);

    substring(str, pos, len);

    return 0;

}

Output :-



Practical -4

1. Write a program to compare two strings. (A)

#include <stdio.h>

int countLength(char \*name){

    int len=0;

    while (\*name!='\0'){

        len=len+1;

        name++;

    }

    return len;

}

int stringCompare(char \*str1, char \*str2) {

    int i1,i2,flag=0,i=0;

    i1=countLength(str1);

    i2=countLength(str2);

    if(i1!=i2){

        return(0);

    }

    while (i<i1)

    {

        if(\*str1!=\*str2){

            return(0);

        }

        else{

            str1++;

            str2++;

            i++;

            flag=1;

        }

        if(flag==1){

            return(1);

        }

    }

}

int main() {

    char str1[100], str2[100];

    printf("Enter the first string: ");

    gets(str1);

    printf("Enter the second string: ");

    gets(str2);

    int result = stringCompare(str1, str2);

    if (result == 1) {

        printf("The strings are equal.\n");

    } else{

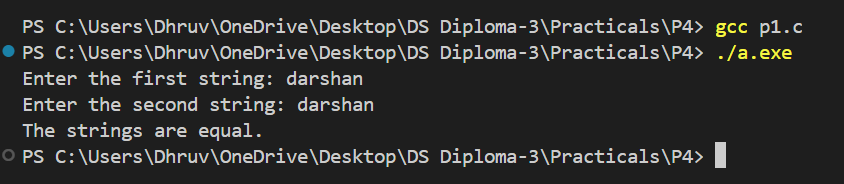
        printf("Both String Are different");

    }

    return 0;

}

Output :-



2. Write a program to display the reverse string of the given string. (A)

#include <stdio.h>

void stringReverse(char \*str1,char \*str2) {

    int i=0;

    while (\*str1!='\0')

    {

        str1++;

        i++;

    }

    str1--;

    while (i>0)

    {

       \*str2=\*str1;

       str1--;

       str2++;

       i--;

    }

    \*str2='\0';

}

int main() {

    char str1[100];

    char str2[100];

    printf("Enter a string: ");

    gets(str1);

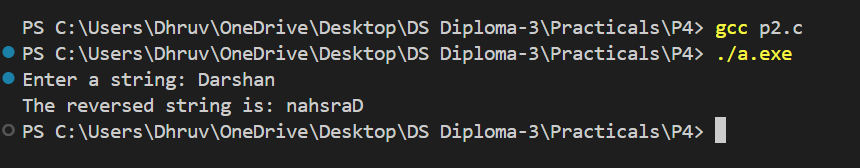
    stringReverse(str1,str2);

    printf("The reversed string is: %s\n", str2);

    return 0;

}

Output :-



3. Write a program to convert given string into uppercase and lowercase. (A)

#include <stdio.h>

void toUppercase(char \*str,char \*str2) {

    while (\*str!='\0') {

        if (\*str >= 'a' && \*str <= 'z') {

            \*str2 = \*str - 32;

        }

        else{

            \*str2=\*str;

        }

        str++;

        str2++;

    }

    \*str2='\0';

}

void toLowercase(char \*str,char \*str2) {

    while (\*str!='\0') {

        if (\*str >= 'A' && \*str <= 'Z') {

            \*str2 = \*str + 32;

        }

        else{

            \*str2=\*str;

        }

        str++;

        str2++;

    }

    \*str2='\0';

}

int main() {

    char str[100];

    char str2[100];

    printf("Enter a string: ");

    gets(str);

    toUppercase(str,str2);

    printf("The string in uppercase is: %s\n", str2);

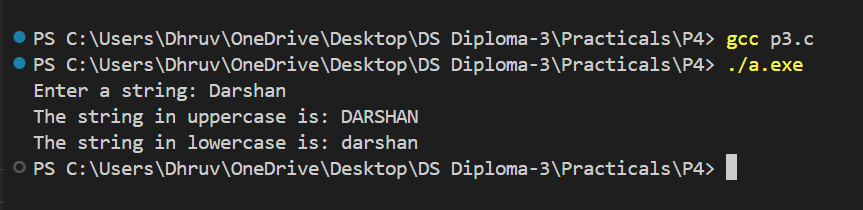
    toLowercase(str,str2);

    printf("The string in lowercase is: %s\n", str2);

    return 0;

}

Output :-



Practical -5

1. Write a menu-driven program to implement the following operations on the Stack using an Array: (A)

i. PUSH, POP

ii. Display all elements of the stack

iii. PEEP, CHANGE

#include<stdio.h>

#define n 5

void push(int s[],int \*top,int x){

    if(\*top>n){

        printf("stack overflow");

        return;

    }else{

        \*top=\*top+1;

        s[\*top]=x;

        printf("-- %d PUSH --",x);

        return;

    }

}

void pop(int s[],int \*top){

    int del;

    if(\*top==-1){

        printf("stack underflow");

        return;

    }else{

        del=s[\*top];

        \*top=\*top-1;

        printf("-- %d POP --",del);

        return;

    }

}

int peep(int s[],int \*top,int i){

    if(\*top<0){

        printf("stack empty");

        return -1;

    }if(top-i+1<0){

        printf("Stack underflow");

        return -1;

    }

    int value = s[\*top-i+1];

    printf("element at position %d from top is %d",i,value);

}

void change(int s[],int \*top,int i,int x){

    if(\*top<0){

        printf("stack empty");

        return;

    }if(top-i+1<0){

        printf("Stack underflow");

        return;

    }

    s[\*top-i+1]=x;

}

void display(int s[],int \*top){

    int i = \*top;

    if(\*top<0){

        printf("stack underflow");

        return;

    }

    else{

        printf("--- stack ---\n");

        while(i>=0){

            printf("%d\n",s[i]);

            i--;

        }

    }

}

void main(){

    int x,po,choice,top=-1,s[100];

    do{

        printf("\n1.PUSH\n2.POP\n3.PEEP\n4.CHNAGE\n5.DISPLAY\n99.EXIT\nEnter Your choice : ");

        scanf("%d",&choice);

        switch(choice){

            case 1:

                printf("Enter element : ");

                scanf("%d",&x);

                push(s,&top,x);

                printf("\ntop-%d",top);

                break;

            case 2:

                pop(s,&top);

                break;

            case 3:

                printf("Enter position of element : ");

                scanf("%d",&po);

                peep(s,&top,po);

                break;

            case 4:

                printf("Enter element : ");

                scanf("%d",&x);

                printf("Enter position of element to be change: ");

                scanf("%d",&po);

                change(s,&top,po,x);

                break;

            case 5:

                display(s,&top);

                break;

            default:

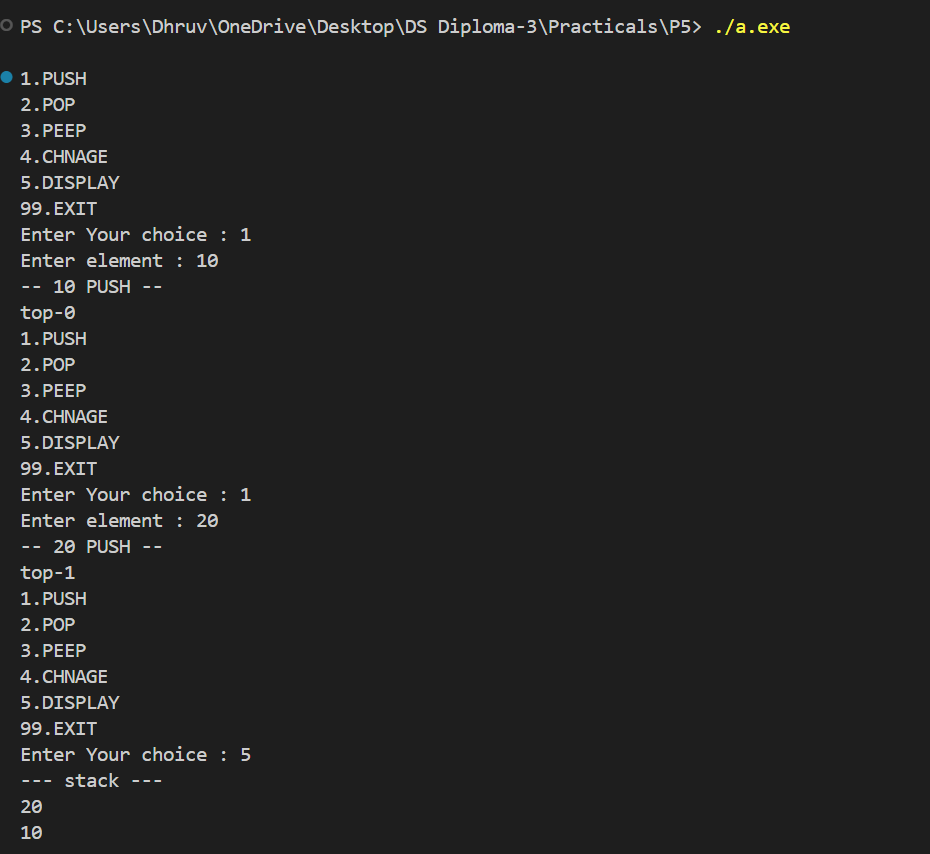
                printf("### Invalid choice ###");

        }

    }while(choice!=99);

}

Output :-



2. Write a program to calculate the average value of the Stack elements. (B)

//2. Write a program to calculate the average value of the Stack elements. (B)

#include<stdio.h>

#define n 5

void push(int s[],int \*top,int x){

    if(\*top>=n){

        printf("stack overflow");

        return;

    }else{

        \*top=\*top+1;

        s[\*top]=x;

        return;

    }

}

void pop(int s[],int \*top){

    int del;

    if(\*top==-1){

        printf("stack underflow");

        return;

    }else{

        del=s[\*top];

        \*top=\*top-1;

        return;

    }

}

float avgOfStack(int s[],int \*top){

    int i,sum=0;

    for(i=0;i<=\*top;i++){

        sum+=s[i];

    }

    return sum / (\*top+1);

}

void main(){

    int i,s[100],top=-1;

    push(s,&top,55);

    push(s,&top,52);

    push(s,&top,68);

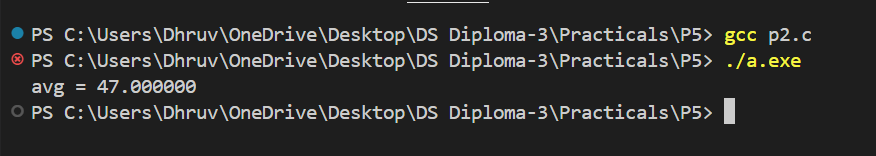
    push(s,&top,16);

    float avg = avgOfStack(s,&top);

    printf("avg = %f",avg);

}

Output :-



3. Write a program to ƒind the minimum and maximum element from a Stack (B).

#include<stdio.h>

#define n 5

void push(int s[],int \*top,int x){

    if(\*top>=n){

        printf("stack overflow");

        return;

    }else{

        \*top=\*top+1;

        s[\*top]=x;

        return;

    }

}

void pop(int s[],int \*top){

    int del;

    if(\*top==-1){

        printf("stack underflow");

        return;

    }else{

        del=s[\*top];

        \*top=\*top-1;

        return;

    }

}

int minimum(int s[],int \*top){

    int i,min=s[0];

    for(i=1;i<=\*top;i++){

        if(min > s[i])

            min=s[i];

    }

    return min;

}

int maximum(int s[],int \*top){

    int i,max=s[0];

    for(i=1;i<=\*top;i++){

        if(max < s[i])

            max=s[i];

    }

    return max;

}

void main(){

    int i,s[100],top=-1;

    push(s,&top,55);

    push(s,&top,52);

    push(s,&top,68);

    push(s,&top,16);

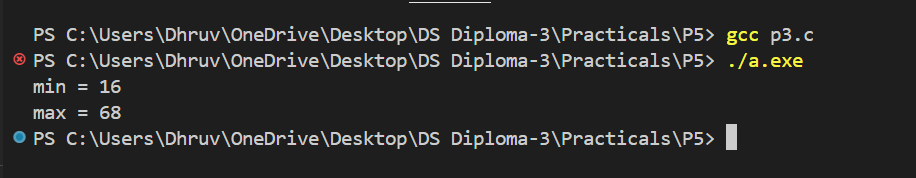
    int min = minimum(s,&top);

    int max = maximum(s,&top);

    printf("min = %d\nmax = %d",min,max);

}

Output :-



4. Write a program for the evaluation of postƒix Expression using Stack. (C)

#include<stdio.h>

void push(int s[],int \*top,int x,int n){

    if(\*top>=n){

        printf("stack overflow");

        return;

    }else{

        \*top=\*top+1;

        s[\*top]=x;

        return;

    }

}

int pop(int s[],int \*top){

    int n = sizeof(s)/sizeof(s[0]);

    int del;

    if(\*top==-1){

        printf("stack underflow");

        return -1;

    }else{

        del=s[\*top];

        \*top=\*top-1;

        return del;

    }

}

int performOperation(int op1,int op2,char operater){

    switch(operater){

        case '+':

            return op1+op2;

            break;

        case '-':

            return op1-op2;

            break;

        case '\*':

            return op1\*op2;

            break;

        case '/':

            return op1/op2;

            break;

        default:

            printf("invalid operater");

    }

}

void postfixEvalution(char postfix[]){

    int top=-1,value=-1,s[100],n=100,i;

    for(i=0;postfix[i]!='\0';i++){

        if(postfix[i]>='0' && postfix[i]<='9'){

            push(s,&top,postfix[i]-'0',n);

        }

        else{

            int op2 = pop(s,&top);

            int op1 = pop(s,&top);

            value = performOperation(op1,op2,postfix[i]);

            push(s,&top,value,n);

        }

    }

    printf("\n%d",pop(s,&top));

}

void main(){

    char postfix[100];

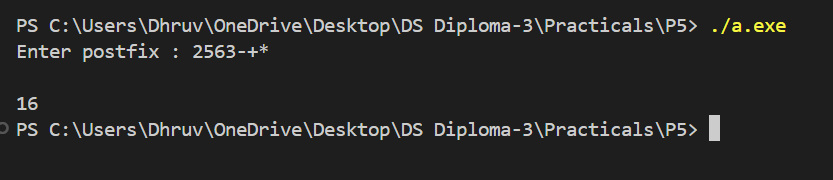
    printf("Enter postfix : ");

    gets(postfix);

    postfixEvalution(postfix);

}

Output :-



5. Write a program for the evaluation of preƒix Expression using Stack. (C)

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#define size 100

int s[size], top = -1;

void push(int val)

{

    if (top == size - 1)

        printf("Stack Overflow\n\n");

    else

        s[++top] = val;

}

int pop()

{

    if (top == -1)

    {

        printf("Stack is Underflow\n");

        return 0;

    }

    else

        return s[top--];

}

double calculate(char c, double a, double b)

{

    switch (c)

    {

    case '+':

        return (a + b);

    case '-':

        return (a - b);

    case '\*':

        return (a \* b);

    case '/':

        if (b == 0)

        {

            printf("Division by Zero\n");

            exit(EXIT\_FAILURE);

        }

        return (a / b);

    default:

        return 0;

    }

}

double evaluate(const char \*expression)

{

    int len = strlen(expression);

    int i,j;

    for (i = len - 1; i >= 0; i--)

    {

        char c = expression[i];

        if (c == ' ')

        {

            continue;

        }

        else if (isdigit(c))

        {

            int num = 0;

            char temp[size] = "";

            int tempIndex = 0;

            while (i >= 0 && isdigit(expression[i]))

            {

                temp[tempIndex++] = expression[i--];

            }

            i++;

            char str[size] = "";

            int strIndex = 0;

            for (j = tempIndex - 1; j >= 0; j--)

            {

                str[strIndex++] = temp[j];

            }

            str[strIndex] = '\0';

            sscanf(str, "%d", &num);

            push(num);

        }

        else

        {

            double op1 = pop();

            double op2 = pop();

            push(calculate(c, op1, op2));

        }

    }

    return pop();

}

void main()

{

    char ex[size];

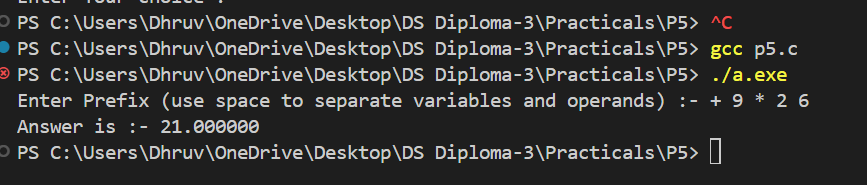
    printf("Enter Prefix: ");

    gets(ex);

    printf("Answer is : %lf\n", evaluate(ex));

}

Output :-



Practical -6

1. Write a menu-driven program to implement the following operations on the Queue using an Array: (A)

i. ENQUEUE

ii. DEQUEUE

iii. Display all elements of the queue

#include<stdio.h>

#define size 2

void enqueue(int q[],int \*f,int \*r,int x){

    if(\*r>=size){

        printf("queue overflow");

        return;

    }

    \*r=\*r+1;

    q[\*r]=x;

    if(\*f==-1)

        \*f=0;

}

void dequeue(int q[],int \*f,int \*r,int \*x){

    if(\*f==-1){

        printf("queue is underflow");

        return;

    }

    \*x=q[\*f];

    printf("\n%d delete from queue",\*x);

    if(\*r==\*f){

        \*r=-1;

        \*f=-1;

    }else{

        \*f=\*f+1;

    }

}

void display(int q[],int \*f,int \*r){

    int i;

    if(\*f==-1 && \*r==-1){

        printf("queue is underflow");

        return;

    }

    for(i=\*f;i<=\*r;i++){

        printf("%d ",q[i]);

    }

}

void main(){

    int q[100],r=-1,f=-1,choice,x;

    do{

        printf("\n1.ENQUEUE\n2.DEQUEUE\n3.DISPLAY\n99.EXIT\nEnter Your choice : ");

        scanf("%d",&choice);

        switch(choice){

            case 1:

                printf("Enter element : ");

                scanf("%d",&x);

                enqueue(q,&f,&r,x);

                break;

            case 2:

                dequeue(q,&f,&r,&x);

                break;

            case 3:

                display(q,&f,&r);

                break;

            default:

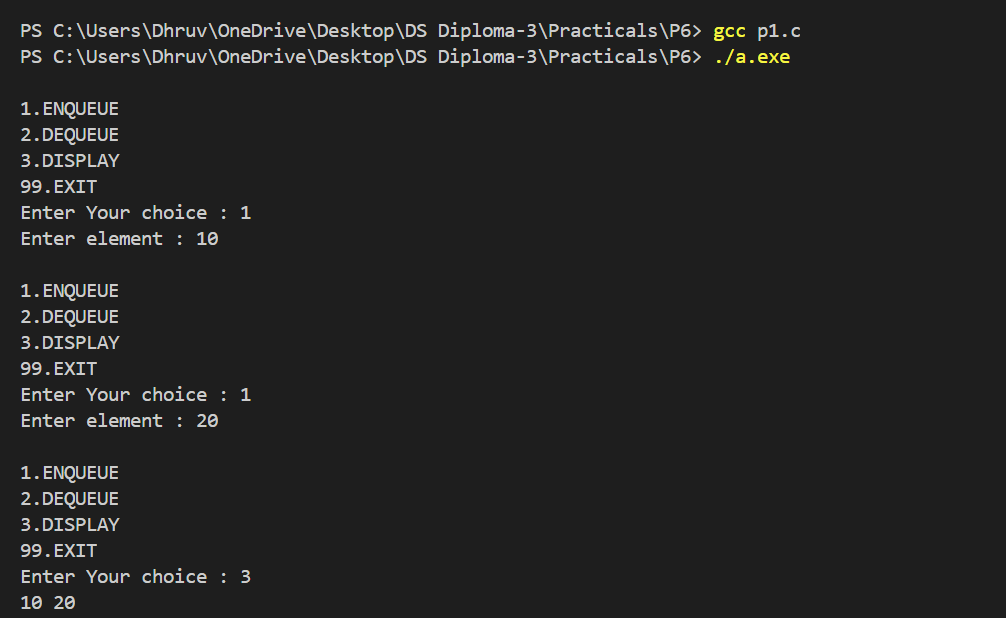
                printf("### Invalid choice ###");

        }

    }while(choice!=99);

}

Output :-



2. Write a program to calculate the average value of the Simple Queue elements.(B)

#include<stdio.h>

#define size 5

void enqueue(int q[],int \*f,int \*r,int x){

    if(\*r>=size){

        printf("queue overflow");

        return;

    }

    \*r=\*r+1;

    q[\*r]=x;

    if(\*f==-1)

        \*f=0;

}

void dequeue(int q[],int \*f,int \*r,int \*x){

    if(\*f==-1){

        printf("queue is underflow");

        return;

    }

    \*x=q[\*f];

    printf("\n%d delete from queue",\*x);

    if(\*r==\*f){

        \*r=-1;

        \*f=-1;

    }else{

        \*f=\*f+1;

    }

}

float avgOfQueue(int q[],int \*f,int \*r){

    int i,sum=0;

    for(i=\*f;i<=\*r;i++){

        sum=sum+q[i];

    }

    return (float)sum/(\*r-\*f+1);

}

void main(){

    int q[100],r=-1,f=-1,avg;

    enqueue(q,&f,&r,56);

    enqueue(q,&f,&r,77);

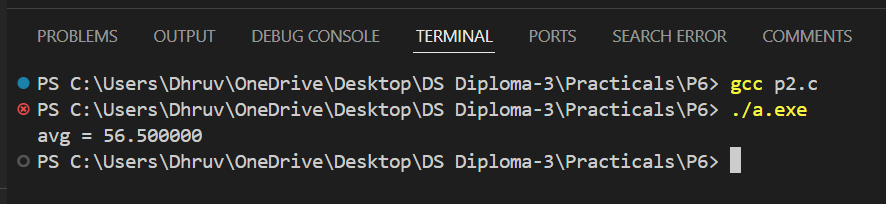
    enqueue(q,&f,&r,48);

    enqueue(q,&f,&r,45);

    printf("avg = %f",avgOfQueue(q,&f,&r));

}

Output :-



3. Write a program to ƒnd the minimum and maximum elements from Simple Queue. (B)

#include<stdio.h>

#define size 5

void enqueue(int q[],int \*f,int \*r,int x){

    if(\*r>=size){

        printf("queue overflow");

        return;

    }

    \*r=\*r+1;

    q[\*r]=x;

    if(\*f==-1)

        \*f=0;

}

void dequeue(int q[],int \*f,int \*r,int \*x){

    if(\*f==-1){

        printf("queue is underflow");

        return;

    }

    \*x=q[\*f];

    printf("\n%d delete from queue",\*x);

    if(\*r==\*f){

        \*r=-1;

        \*f=-1;

    }else{

        \*f=\*f+1;

    }

}

int minimum(int q[],int \*f,int \*r){

    int i,min=q[0];

    for(i=\*f+1;i<=\*r;i++){

        if(min>q[i])

            min=q[i];

    }

    return min;

}

int maximum(int q[],int \*f,int \*r){

    int i,max=q[0];

    for(i=\*f+1;i<=\*r;i++){

        if(max<q[i])

            max=q[i];

    }

    return max;

}

void main(){

    int q[100],r=-1,f=-1,n=100,avg;

    enqueue(q,&f,&r,56);

    enqueue(q,&f,&r,77);

    enqueue(q,&f,&r,48);

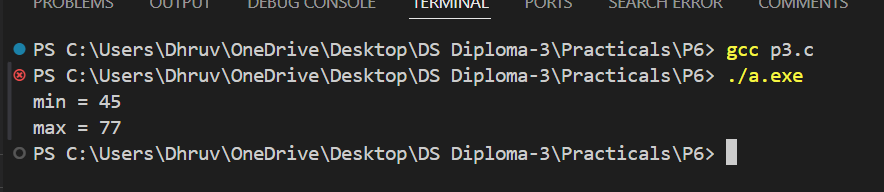
    enqueue(q,&f,&r,45);

    printf("min = %d",minimum(q,&f,&r));

    printf("\nmax = %d",maximum(q,&f,&r));

}

Output :-



4. Write a program to sort the elements of a Simple Queue in ascending order. (C)

#include<stdio.h>

#define size 5

void enqueue(int q[],int \*f,int \*r,int x){

    if(\*r>=size){

        printf("queue overflow");

        return;

    }

    \*r=\*r+1;

    q[\*r]=x;

    if(\*f==-1)

        \*f=0;

}

void dequeue(int q[],int \*f,int \*r,int \*x){

    if(\*f==-1){

        printf("queue is underflow");

        return;

    }

    \*x=q[\*f];

    if(\*r==\*f){

        \*r=-1;

        \*f=-1;

    }else{

        \*f=\*f+1;

    }

}

void sortQueue(int q[], int \*f, int \*r) {

    if (\*f == -1) {

        printf("Queue is empty, nothing to sort\n");

        return;

    }

    int temp[size],count = 0,i,j;

    while (\*f != -1) {

        int x=0;

        dequeue(q, f, r,&x);

        temp[count] = x;

        count++;

    }

    for (i = 0; i < count - 1; i++) {

        for (j = 0; j < count - i - 1; j++) {

            if (temp[j] > temp[j + 1]) {

                int swap = temp[j];

                temp[j] = temp[j + 1];

                temp[j + 1] = swap;

            }

        }

    }

    \*f = -1;

    \*r = -1;

    for (i = 0; i < count; i++) {

        enqueue(q, f, r, temp[i]);

    }

}

void main(){

    int q[100],r=-1,f=-1,i;

    enqueue(q,&f,&r,56);

    enqueue(q,&f,&r,77);

    enqueue(q,&f,&r,48);

    enqueue(q,&f,&r,45);

    printf("\nbefore sort queue");

    for(i=f;i<=r;i++){

        printf("%d ",q[i]);

    }

    printf("\nafter sort queue");

    sortQueue(q,&f,&r);

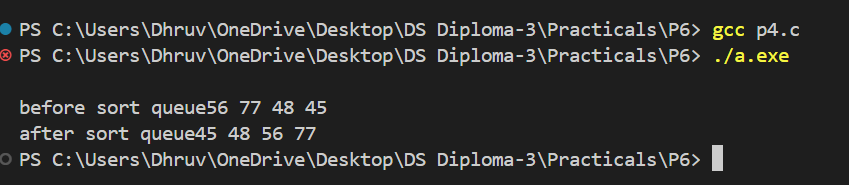
    for(i=f;i<=r;i++){

        printf("%d ",q[i]);

    }

}

Output :-



Practical -7

1. Write a menu-driven program to implement the following operations on a circular queue using an Array: (A)

i. ENQUEUE

ii. DEQUEUE

iii. Display all elements of the circular queue

#include<stdio.h>

#define size 5

void enqueue(int q[],int \*f,int \*r,int x){

    if((\*f==0 && \*r==size-1) || (\*f==\*r+1)){

        printf("circular queue overflow");

        return;

    }else if(\*f==-1 && \*r==-1){

        \*f=0;

        \*r=0;

    }else if(\*f!=0 && \*r==size-1){

        \*r=0;

    }else{

        \*r=\*r+1;

    }

    q[\*r]=x;

}

void dequeue(int q[],int \*f,int \*r,int \*x){

    if(\*f==-1){

        printf("circular queue is underflow");

        return;

    }

    \*x=q[\*f];

    printf("\n%d delete from queue",\*x);

    if(\*r==\*f){

        \*r=-1;

        \*f=-1;

    }else if(\*f==size-1){

        \*f=0;

    }

    else{

        \*f=\*f+1;

    }

}

void display(int q[],int \*f,int \*r){

    int i;

    if(\*f==-1 && \*r==-1){

        printf("queue is underflow");

        return;

    }

    if(\*f<=\*r){

        for(i=\*f;i<=\*r;i++){

            printf("%d ",q[i]);

        }

    }else{

        for(i=\*f;i<size;i++){

            printf("%d ",q[i]);

        }

        for(i=0;i<=\*r;i++){

            printf("%d ",q[i]);

        }

    }

}

void main(){

    int q[100],r=-1,f=-1,choice,x;

    do{

        printf("\n1.ENQUEUE\n2.DEQUEUE\n3.DISPLAY\n99.EXIT\nEnter Your choice : ");

        scanf("%d",&choice);

        switch(choice){

            case 1:

                printf("Enter element : ");

                scanf("%d",&x);

                enqueue(q,&f,&r,x);

                break;

            case 2:

                dequeue(q,&f,&r,&x);

                break;

            case 3:

                display(q,&f,&r);

                break;

            default:

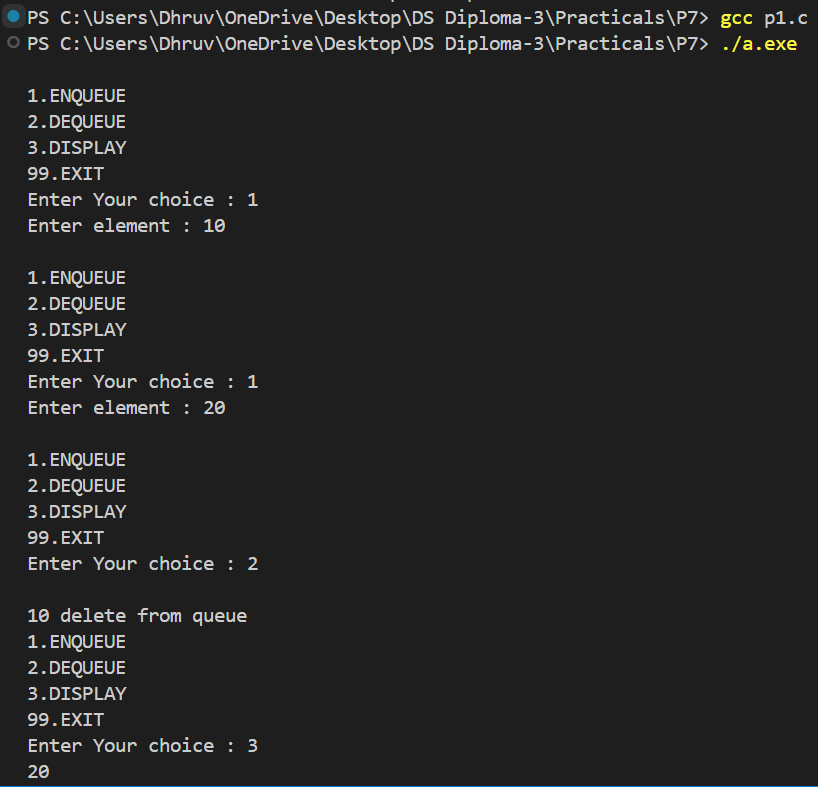
                printf("### Invalid choice ###");

        }

    }while(choice!=99);

}

Output :-



2. Write a program to calculate the average value of the Circular Queue elements. (B)

#include<stdio.h>

#define size 5

void enqueue(int q[],int \*f,int \*r,int x){

    if((\*f==0 && \*r==size-1) || (\*f==\*r+1)){

        printf("circular queue overflow");

        return;

    }else if(\*f==-1 && \*r==-1){

        \*f=0;

        \*r=0;

    }else if(\*f!=0 && \*r==size-1){

        \*r=0;

    }else{

        \*r=\*r+1;

    }

    q[\*r]=x;

}

void dequeue(int q[],int \*f,int \*r,int \*x){

    if(\*f==-1){

        printf("circular queue is underflow");

        return;

    }

    \*x=q[\*f];

    if(\*r==\*f){

        \*r=-1;

        \*f=-1;

    }else if(\*f==size-1){

        \*f=0;

    }

    else{

        \*f=\*f+1;

    }

}

float avg(int q[],int \*f,int \*r){

    int i,sum=0,totalel=0;

    if(\*f==-1 && \*r==-1){

        printf("queue is underflow");

        return -1;

    }

    if(\*f<=\*r){

        for(i=\*f;i<=\*r;i++){

            sum+=q[i];

            totalel++;

        }

        return (float)sum/totalel;

    }else{

        for(i=\*f;i<size;i++){

            sum+=q[i];

            totalel++;

        }

        for(i=0;i<=\*r;i++){

            sum+=q[i];

            totalel++;

        }

        return (float)sum/totalel;

    }

}

void main(){

    int q[100],r=-1,f=-1,choice,x;

    enqueue(q,&f,&r,11);

    enqueue(q,&f,&r,12);

    enqueue(q,&f,&r,13);

    enqueue(q,&f,&r,14);

    enqueue(q,&f,&r,15);

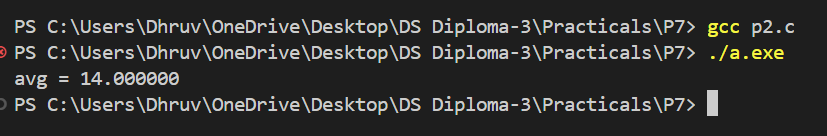
    dequeue(q,&f,&r,&x);

    enqueue(q,&f,&r,16);

    printf("avg = %f",avg(q,&f,&r));

}

Output :-



3. Write a program to ƒind the minimum and maximum elements from the Circular Queue. (B)

#include<stdio.h>

#define size 5

void enqueue(int q[],int \*f,int \*r,int x){

    if((\*f==0 && \*r==size-1) || (\*f==\*r+1)){

        printf("circular queue overflow");

        return;

    }else if(\*f==-1 && \*r==-1){

        \*f=0;

        \*r=0;

    }else if(\*f!=0 && \*r==size-1){

        \*r=0;

    }else{

        \*r=\*r+1;

    }

    q[\*r]=x;

}

void dequeue(int q[],int \*f,int \*r,int \*x){

    if(\*f==-1){

        printf("circular queue is underflow");

        return;

    }

    \*x=q[\*f];

    if(\*r==\*f){

        \*r=-1;

        \*f=-1;

    }else if(\*f==size-1){

        \*f=0;

    }

    else{

        \*f=\*f+1;

    }

}

int min(int q[],int \*f,int \*r){

    int i,min=q[\*f],totalel;

    if(\*f==-1 && \*r==-1){

        printf("queue is underflow");

        return -1;

    }

    if(\*f<=\*r){

        for(i=\*f+1;i<=\*r;i++){

            if(q[i]<min)

                min=q[i];

        }

        return min;

    }else{

        for(i=\*f;i<size;i++){

            if(q[i]<min)

                min=q[i];

        }

        for(i=0;i<=\*r;i++){

            if(q[i]<min)

                min=q[i];

        }

        return min;

    }

}

int max(int q[],int \*f,int \*r){

    int i,max=q[\*f],totalel;

    if(\*f==-1 && \*r==-1){

        printf("queue is underflow");

        return -1;

    }

    if(\*f<=\*r){

        for(i=\*f+1;i<=\*r;i++){

            if(q[i]>max)

                max=q[i];

        }

        return max;

    }else{

        for(i=\*f;i<size;i++){

            if(q[i]>max)

                max=q[i];

        }

        for(i=0;i<=\*r;i++){

            if(q[i]>max)

                max=q[i];

        }

        return max;

    }

}

void main(){

    int q[100],r=-1,f=-1,choice,x;

    enqueue(q,&f,&r,11);

    enqueue(q,&f,&r,12);

    enqueue(q,&f,&r,13);

    enqueue(q,&f,&r,14);

    enqueue(q,&f,&r,15);

    dequeue(q,&f,&r,&x);

    dequeue(q,&f,&r,&x);

    enqueue(q,&f,&r,16);

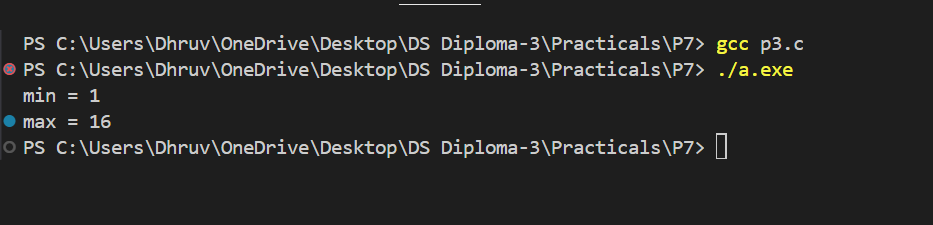
    enqueue(q,&f,&r,1);

    printf("min = %d",min(q,&f,&r));

    printf("\nmax = %d",max(q,&f,&r));

}

Output :-



4. Write a program to sort the elements of a Circular Queue in descending order. (C)

#include<stdio.h>

#define size 5

void enqueue(int q[],int \*f,int \*r,int x){

    if((\*f==0 && \*r==size-1) || (\*f==\*r+1)){

        printf("circular queue overflow");

        return;

    }else if(\*f==-1 && \*r==-1){

        \*f=0;

        \*r=0;

    }else if(\*f!=0 && \*r==size-1){

        \*r=0;

    }else{

        \*r=\*r+1;

    }

    q[\*r]=x;

}

void dequeue(int q[],int \*f,int \*r,int \*x){

    if(\*f==-1){

        printf("circular queue is underflow");

        return;

    }

    \*x=q[\*f];

    if(\*r==\*f){

        \*r=-1;

        \*f=-1;

    }else if(\*f==size-1){

        \*f=0;

    }

    else{

        \*f=\*f+1;

    }

}

void sort(int q[], int \*f, int \*r) {

    if (\*f == -1) {

        printf("Queue is empty\n");

        return;

    }

    int temp[size],count = 0,i,j;

    while (\*f != -1) {

        int x=-1;

        dequeue(q, f, r, &x);

        temp[count] = x;

        count++;

    }

    for (i = 0; i < count - 1; i++) {

        for (j = 0; j < count - i - 1; j++) {

            if (temp[j] > temp[j + 1]) {

                int swap = temp[j];

                temp[j] = temp[j + 1];

                temp[j + 1] = swap;

            }

        }

    }

    for (i = 0; i < count; i++) {

        enqueue(q, f, r, temp[i]);

    }

}

void display(int q[],int \*f,int \*r){

    int i;

    if(\*f==-1 && \*r==-1){

        printf("quesu is underflow");

        return;

    }

    if(\*f<=\*r){

        for(i=\*f;i<=\*r;i++){

            printf("%d ",q[i]);

        }

    }else{

        for(i=\*f;i<size;i++){

            printf("%d ",q[i]);

        }

        for(i=0;i<=\*r;i++){

            printf("%d ",q[i]);

        }

    }

}

void main(){

    int q[100],r=-1,f=-1,choice,x,i;

    enqueue(q,&f,&r,17);

    enqueue(q,&f,&r,12);

    enqueue(q,&f,&r,23);

    enqueue(q,&f,&r,14);

    enqueue(q,&f,&r,5);

    dequeue(q,&f,&r,&x);

    dequeue(q,&f,&r,&x);

    enqueue(q,&f,&r,16);

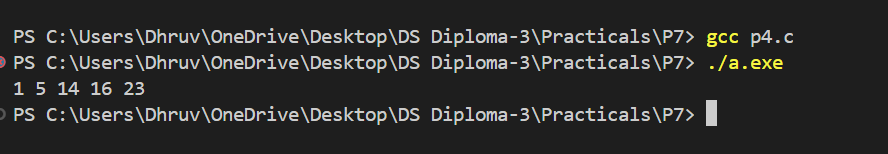
    enqueue(q,&f,&r,1);

    sort(q,&f,&r);

    display(q,&f,&r);

}

Output :-



Practical -8

1. Write a program to implement a node structure for a singly linked list. Read the data in a node, and print the node. (A)

#include<stdio.h>

#include<stdlib.h>

struct Node{

    int info;

    struct Node\* link;

};

struct Node\* createNode(int data){

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

    newNode->info = data;

    newNode->link = NULL;

    return newNode;

}

void displayNode(struct Node\* node){

    printf("data : %d",node->info);

}

void main(){

    int data;

    printf("Enter node data : ");

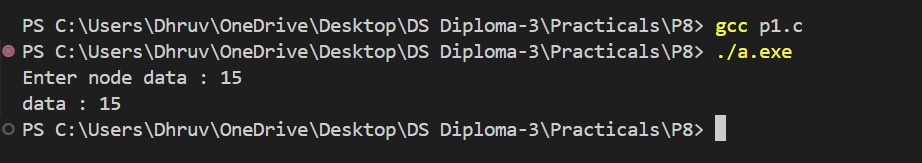
    scanf("%d",&data);

    struct Node\* newNode = createNode(data);

    displayNode(newNode);

}

Output :-



2. Write a menu-driven program to implement the following operations on the singly linked list: (A)

i. Insert a node at the beginning of the linked list.

ii. Insert a node at the end of the linked list.

iii. Delete a ƒirst node of the linked list.

iv. Delete a last node of the linked list.

v. Display all nodes.

#include<stdio.h>

#include<stdlib.h>

struct Node{

    int info;

    struct Node\* link;

};

void InsertAtBeg(int data, struct Node\*\* first) {

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

    if (newNode == NULL) {

        printf("Memory allocation failed\n");

        return;

    }

    newNode->info = data;

    newNode->link = \*first;

    \*first = newNode;

}

void InsertAtEnd(int data,struct Node\*\* first){

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

    if (newNode == NULL) {

        printf("Memory allocation failed\n");

        return;

    }

    newNode->info = data;

    newNode->link = NULL;

    if(\*first == NULL){

        \*first = newNode;

    }else{

        struct Node\* ptr = \*first;

        while(ptr->link != NULL){

            ptr = ptr->link;

        }

        ptr->link = newNode;

    }

}

void deleteFirst(struct Node\*\* first){

    if(\*first == NULL){

        printf("Linked list is empty");

        return;

    }

    if((\*first)->link == NULL){

        free(\*first);

        \*first = NULL;

    }else{

        \*first = (\*first)->link;

    }

}

void deleteEnd(struct Node\*\* first){

    if(\*first == NULL){

        printf("Linked list is empty");

        return;

    }

    if((\*first)->link == NULL){

        free(\*first);

        \*first = NULL;

    }else{

        struct Node\* ptr = \*first;

        struct Node\* preptr = \*first;

        while(ptr->link != NULL){

            preptr = ptr;

            ptr=ptr->link;

        }

        free(preptr->link);

        preptr->link=NULL;

    }

}

void display(struct Node\* first){

    if (first == NULL) {

        printf("The list is empty.\n");

        return;

    }

    struct Node\* ptr = first;

    while (ptr != NULL) {

        printf("%d -> ", ptr->info);

        ptr = ptr->link;

    }

    printf("NULL\n");

}

void main(){

    struct Node\* first = NULL;

    int choice=-1,x;

    do{

        printf("\n1.Insert at the beginning\n2.Insert at the end\n3.Delete a first node\n4.Delete a last node\n5.Display all nodes.\nEnter your choice : ");

        scanf("%d",&choice);

        switch(choice){

            case 1:

                printf("enter data to be insert : ");

                scanf("%d",&x);

                InsertAtBeg(x,&first);

                break;

            case 2:

                printf("enter data to be insert : ");

                scanf("%d",&x);

                InsertAtEnd(x,&first);

                break;

            case 3:

                deleteFirst(&first);

                break;

            case 4:

                deleteEnd(&first);

                break;

            case 5:

                display(first);

                break;

            default:

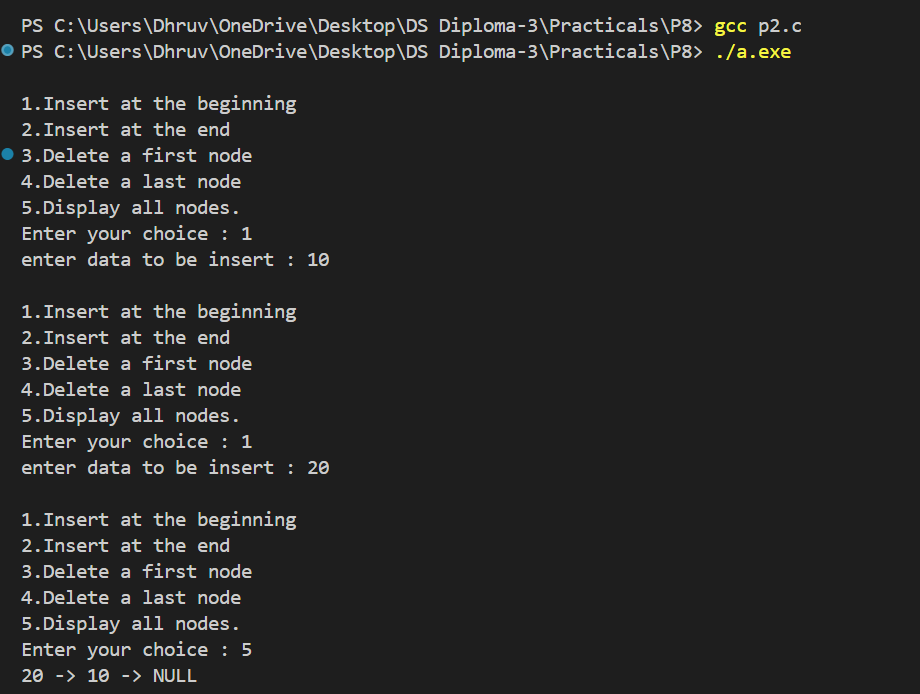
                printf("### Invalid choice ###");

        }

    }while(choice!=99);

}

Output :-



Practical -9

1. Write a menu-driven program to implement the following operations on the singly linked list: (A)

i. Insert a node before the given node of the linked list.

ii. Insert a node after the given node of the linked list.

iii. Delete a given speciƒic node of the linked list.

iv. Display all nodes.

#include<stdio.h>

#include<stdlib.h>

struct Node{

    int info;

    struct Node\* link;

};

void InsertAtBeg(int data, struct Node\*\* first) {

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

    if (newNode == NULL) {

        printf("Memory allocation failed\n");

        return;

    }

    newNode->info = data;

    newNode->link = \*first;

    \*first = newNode;

}

void insertBeforeNode(int data,struct Node\*\* first,int n){

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

    if (newNode == NULL) {

        printf("Memory allocation failed\n");

        return;

    }

    newNode->info = data;

    struct Node\* ptr = \*first;

    struct Node\* preptr = \*first;

    while(ptr->info != n){

        preptr = ptr;

        ptr = ptr->link;

    }

    preptr->link = newNode;

    newNode->link = ptr;

}

void insertAfterNode(int data,struct Node\*\* first,int n){

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

    if (newNode == NULL) {

        printf("Memory allocation failed\n");

        return;

    }

    newNode->info = data;

    struct Node\* ptr = \*first;

    struct Node\* preptr = \*first;

    while(preptr->info != n){

        preptr = ptr;

        ptr = ptr->link;

    }

    newNode->link=preptr->link;

    preptr->link=newNode;

}

void deleteSpec(struct Node\*\* first,int n){

    if (\*first == NULL) {

        printf("The list is empty.\n");

        return;

    }

    struct Node\* ptr = \*first;

    struct Node\* preptr = \*first;

    if ((\*first)->info == n) {

        \*first = (\*first)->link;

        free(ptr);

        return;

    }

    while(ptr->info != n){

        preptr = ptr;

        ptr = ptr->link;

    }

    if (ptr == NULL) {

        printf("Node with value %d not found.\n", n);

        return;

    }

    preptr->link = ptr->link;

    free(ptr);

}

void display(struct Node\* first){

    if (first == NULL) {

        printf("The list is empty.\n");

        return;

    }

    struct Node\* ptr = first;

    while (ptr != NULL) {

        printf("%d -> ", ptr->info);

        ptr = ptr->link;

    }

    printf("NULL\n");

}

void main(){

    struct Node\* first = NULL;

    int choice=-1,x,n;

    do{

        printf("\n1.insert at beg\n2.Insert before given node\n3.Insert after given node\n4.Delete specific\n5.Display all nodes.\nEnter your choice : ");

        scanf("%d",&choice);

        switch(choice){

            case 1:

                printf("enter data to be insert : ");

                scanf("%d",&x);

                InsertAtBeg(x,&first);

                break;

            case 2:

                printf("enter data to be insert : ");

                scanf("%d",&x);

                printf("enter node before insert : ");

                scanf("%d",&n);

                insertBeforeNode(x,&first,n);

                break;

            case 3:

                printf("enter data to be insert : ");

                scanf("%d",&x);

                printf("enter node after insert : ");

                scanf("%d",&n);

                insertAfterNode(x,&first,n);

                break;

            case 4:

                printf("enter node to be delete : ");

                scanf("%d",&n);

                deleteSpec(&first,n);

                break;

            case 5:

                display(first);

                break;

            default:

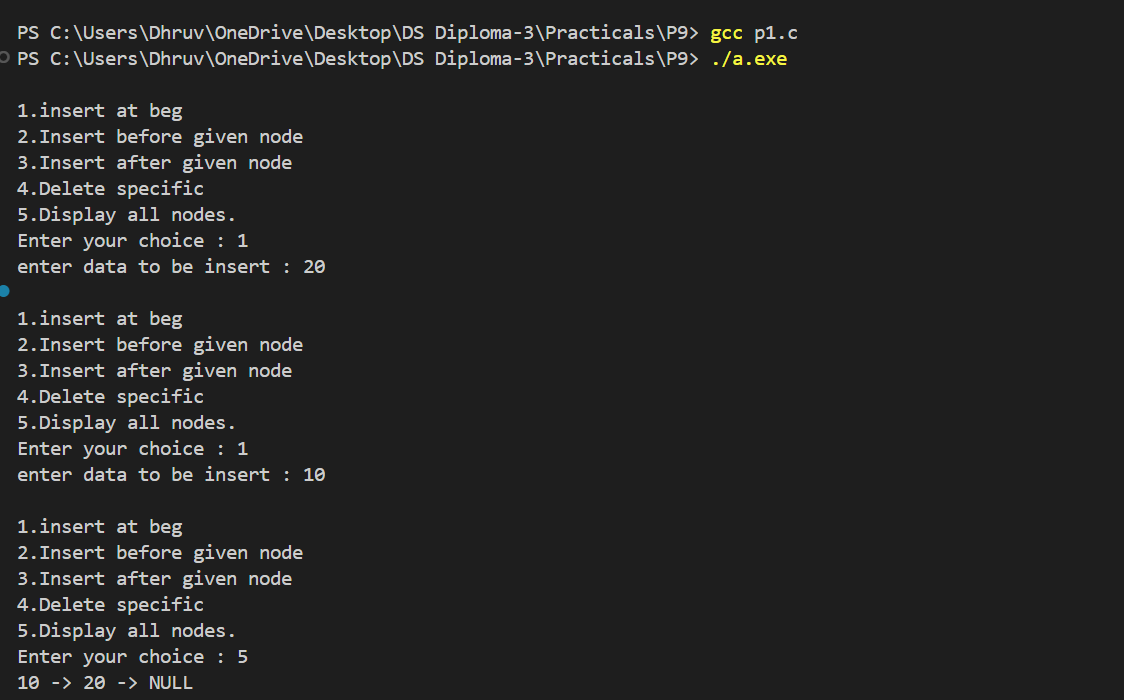
                printf("### Invalid choice ###");

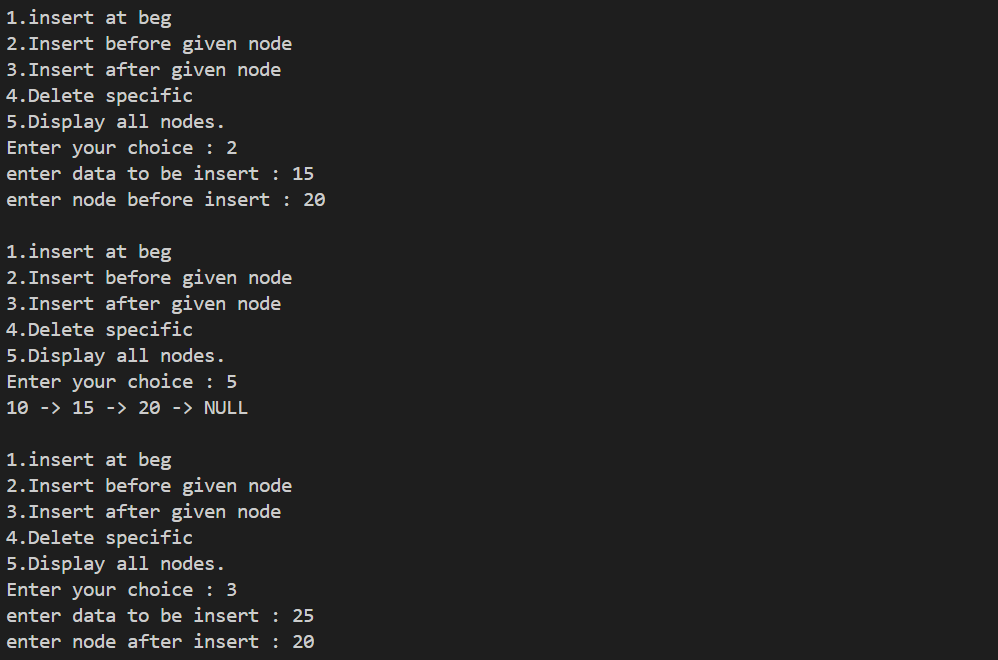
        }

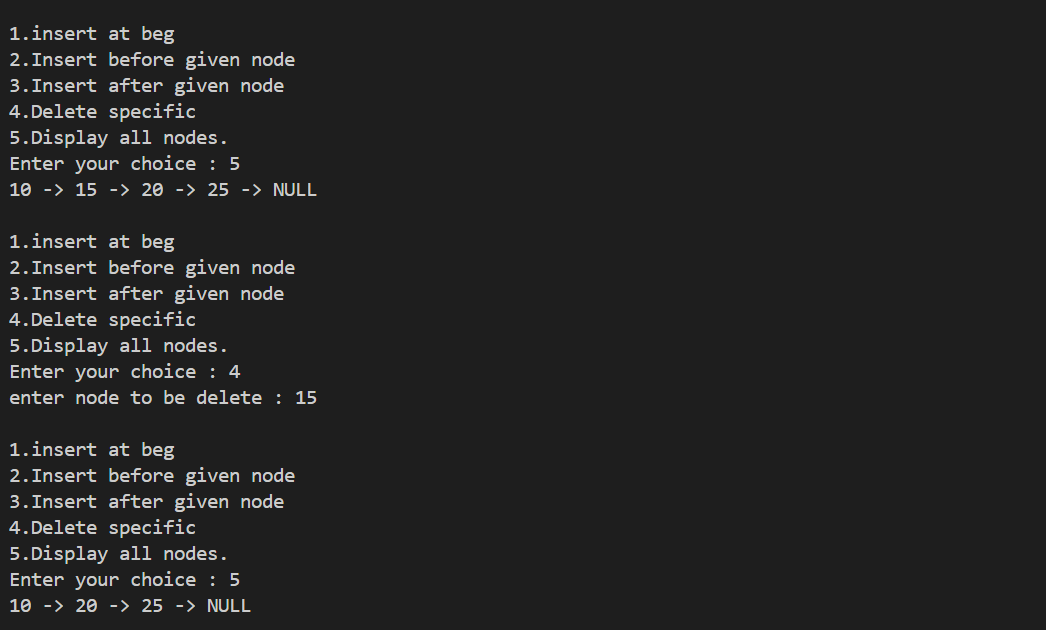
    }while(choice!=99);

}

Output :-







2. Write a program to count the number of nodes in a singly linked list. (A)

#include<stdio.h>

#include<stdlib.h>

struct Node{

    int info;

    struct Node\* link;

};

void InsertAtBeg(int data, struct Node\*\* first) {

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

    if (newNode == NULL) {

        printf("Memory allocation failed\n");

        return;

    }

    newNode->info = data;

    newNode->link = \*first;

    \*first = newNode;

}

int countNode(struct Node\* first){

    int count = 0;

    struct Node\* ptr = first;

    while (ptr != NULL) {

        count++;

        ptr = ptr->link;

    }

    return count;

}void display(struct Node\* first){

    if (first == NULL) {

        printf("The list is empty.\n");

        return;

    }

    struct Node\* ptr = first;

    while (ptr != NULL) {

        printf("%d -> ", ptr->info);

        ptr = ptr->link;

    }

    printf("NULL\n");

}

void main(){

    struct Node\* first = NULL;

    int choice=-1,x,n,len;

    do{

        printf("\n1.Insert at begin\n2.display\n3.count\nEnter your choice : ");

        scanf("%d",&choice);

        switch(choice){

            case 1:

                printf("enter data to be insert : ");

                scanf("%d",&x);

                InsertAtBeg(x,&first);

                break;

            case 2:

                display(first);

                break;

            case 3:

                len = countNode(first);

                printf("length of LL is %d",len);

                break;

            default:

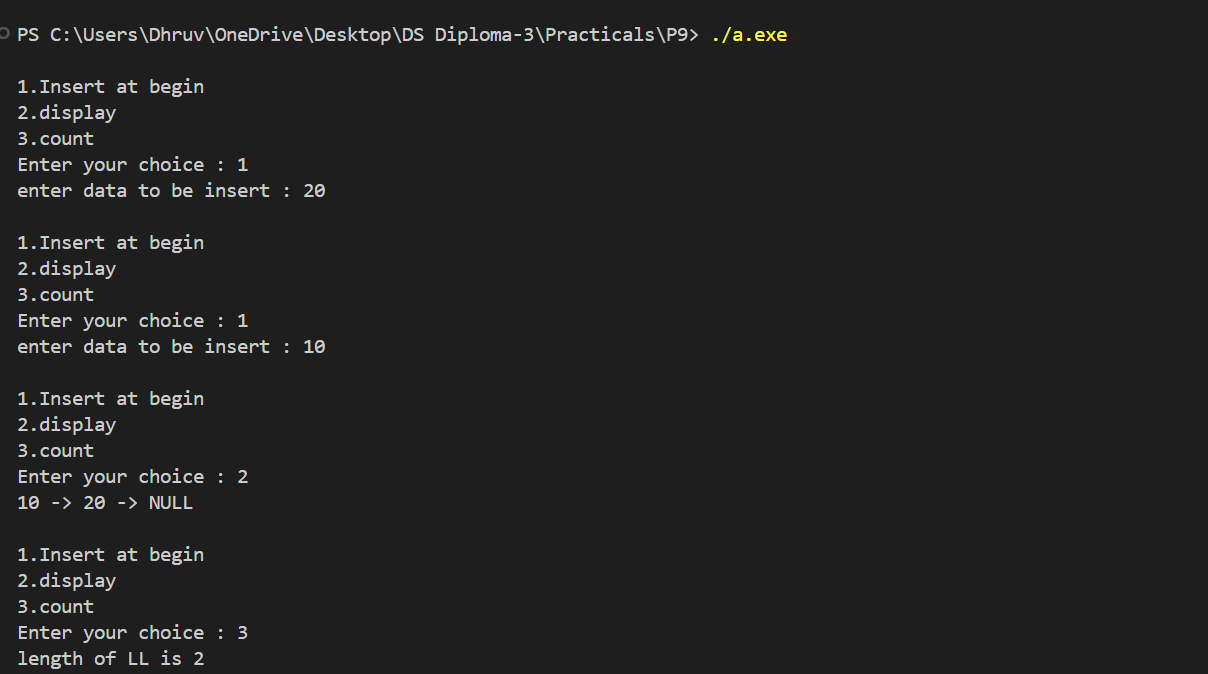
                printf("### Invalid choice ###");

        }

    }while(choice!=99);

}

Output :-



3. Write a program to search given nodes in a singly linked list. (A)

#include<stdio.h>

#include<stdlib.h>

struct Node{

    int info;

    struct Node\* link;

};

void InsertAtBeg(int data, struct Node\*\* first) {

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

    if (newNode == NULL) {

        printf("Memory allocation failed\n");

        return;

    }

    newNode->info = data;

    newNode->link = \*first;

    \*first = newNode;

}

void searchNode(struct Node\* first, int n) {

    if (first == NULL) {

        printf("Linked List is Empty & Underflow\n");

        return;

    }

    int flag = 0;

    struct Node\* ptr = first;

    while (ptr != NULL) {

        if (ptr->info == n) {

            flag = 1;

            break;

        }

        ptr = ptr->link;

    }

    if (flag == 1) {

        printf("Search Successful\n");

    } else {

        printf("Search Unsuccessful\n");

    }

}

void main(){

    struct Node\* first = NULL;

    int choice=-1,x,n,len;

    do{

        printf("\n1.Insert at begi\n2.search node\nEnter your choice : ");

        scanf("%d",&choice);

        switch(choice){

            case 1:

                printf("enter data to be insert : ");

                scanf("%d",&x);

                InsertAtBeg(x,&first);

                break;

            case 2:

                printf("enter data to be search : ");

                scanf("%d",&x);

                searchNode(first,x);

                break;

            default:

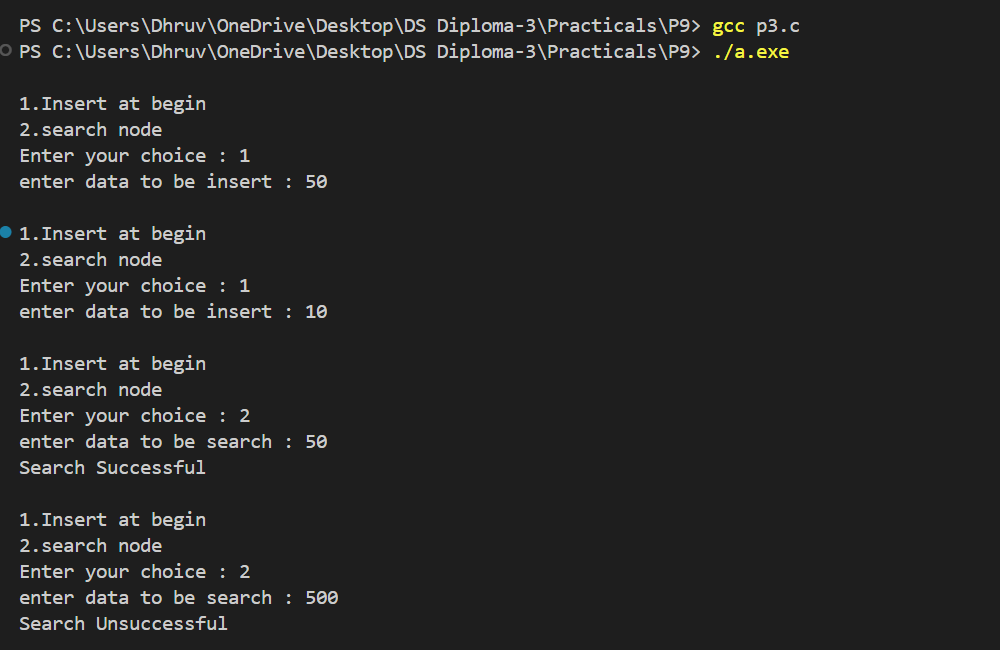
                printf("### Invalid choice ###");

        }

    }while(choice!=99);

}

Output :-



4. Write a program to reverse a linked list. (B)

#include<stdio.h>

#include<stdlib.h>

struct Node{

    int info;

    struct Node\* link;

};

void InsertAtBeg(int data, struct Node\*\* first) {

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

    if (newNode == NULL) {

        printf("Memory allocation failed\n");

        return;

    }

    newNode->info = data;

    newNode->link = \*first;

    \*first = newNode;

}

void rev(struct Node\*\* first) {

    struct Node\* prev = NULL;

    struct Node\* current = \*first;

    struct Node\* next = NULL;

    while (current != NULL) {

        next = current->link;

        current->link = prev;

        prev = current;

        current = next;

    }

    \*first = prev;

    printf("The list has been reversed.\n");

}

void display(struct Node\* first){

    if (first == NULL) {

        printf("The list is empty.\n");

        return;

    }

    struct Node\* ptr = first;

    while (ptr != NULL) {

        printf("%d -> ", ptr->info);

        ptr = ptr->link;

    }

    printf("NULL\n");

}

void main(){

    struct Node\* first = NULL;

    int choice=-1,x,n,len;

    do{

        printf("\n1.Insert before given node\n2.display\n3.reverse LL\nEnter your choice : ");

        scanf("%d",&choice);

        switch(choice){

            case 1:

                printf("enter data to be insert : ");

                scanf("%d",&x);

                InsertAtBeg(x,&first);

                break;

            case 2:

                display(first);

                break;

            case 3:

                rev(&first);

                break;

            default:

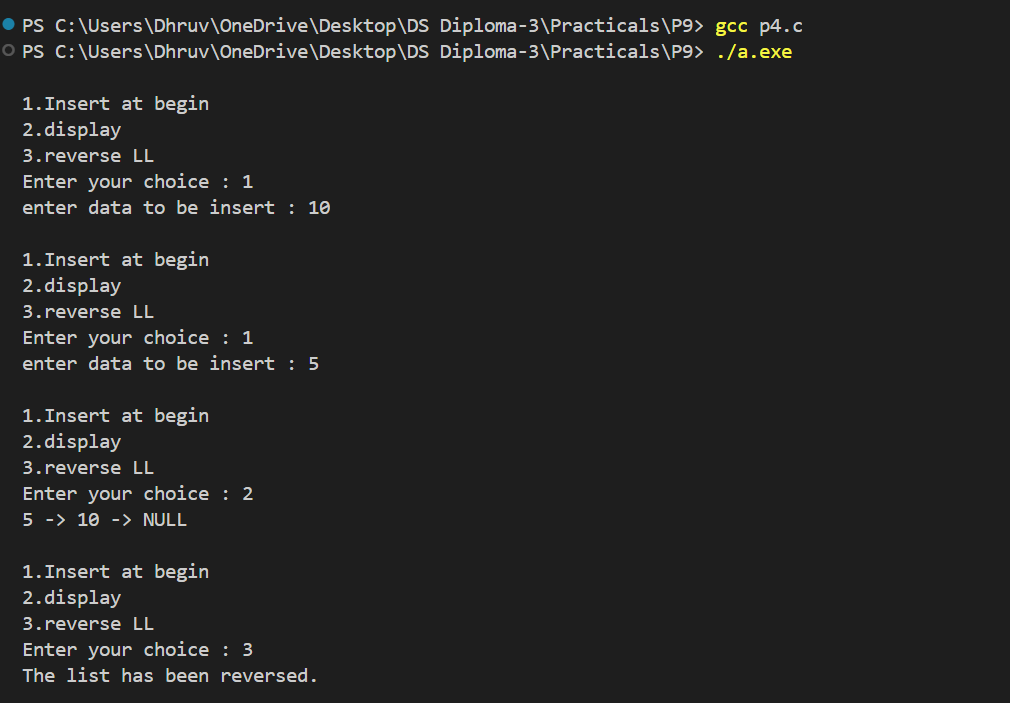
                printf("### Invalid choice ###");

        }

    }while(choice!=99);

}

Output :-





Practical -10

2. Write a menu-driven program to implement Binary Search Tree (BST) & perform the following operations: (B)

i. Insert a node

ii. Delete a node

iii. Search a node

iv. Inorder Traversal

v. Preorder Traversal

vi. Postorder Traversal

#include<stdio.h>

#include<stdlib.h>

struct Node {

    int data;

    struct Node\* left;

    struct Node\* right;

};

struct Node\* insertNode(struct Node\* root, int data) {

    if (root == NULL) {

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

        if (!newNode) {

            printf("Memory allocation failed\n");

            return NULL;

        }

        newNode->data = data;

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

        return newNode;

    }

    if (data < root->data) {

        root->left = insertNode(root->left, data);

    } else if (data > root->data) {

        root->right = insertNode(root->right, data);

    }

    return root;

}

struct Node\* minValueNode(struct Node\* node) {

    struct Node\* temp = node;

    while (temp->left != NULL)

        temp = temp->left;

    return temp;

}

struct Node\* deleteNode(struct Node\* root, int data) {

    if (root == NULL) {

        return root;

    }

    if (data < root->data) {

        root->left = deleteNode(root->left, data);

    } else if (data > root->data) {

        root->right = deleteNode(root->right, data);

    } else {

        if (root->left == NULL) {

            return root->right;

        } else if (root->right == NULL) {

            return root->left;

        }

        struct Node\* temp = minValueNode(root->right);

        root->data = temp->data;

        root->right = deleteNode(root->right, temp->data);

    }

    return root;

}

int searchNode(struct Node\* root, int data) {

    if (root == NULL) {

        printf("Empty Tree\n");

        return NULL;

    }

    if (data == root->data) {

        return root->data;

    }

    if (data < root->data) {

        return searchNode(root->left, data);

    } else {

        return searchNode(root->right, data);

    }

}

void inorderTraversal(struct Node\* root) {

    if (root != NULL) {

        inorderTraversal(root->left);

        printf("%d ", root->data);

        inorderTraversal(root->right);

    }

}

void preorderTraversal(struct Node\* root) {

    if (root != NULL) {

        printf("%d ", root->data);

        preorderTraversal(root->left);

        preorderTraversal(root->right);

    }

}

void postorderTraversal(struct Node\* root) {

    if (root != NULL) {

        postorderTraversal(root->left);

        postorderTraversal(root->right);

        printf("%d ", root->data);

    }

}

void main() {

    struct Node\* root = NULL;

    int choice, data;

    do {

        printf("\n1. Insert a node\n2. Delete a node\n3. Search a node\n4. Inorder traversal\n5. Preorder traversal\n6. Postorder traversal\n7. Exit\nEnter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

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

                scanf("%d", &data);

                root = insertNode(root, data);

                break;

            case 2:

                printf("Enter the data to be deleted: ");

                scanf("%d", &data);

                root = deleteNode(root, data);

                break;

            case 3:

                printf("Enter the data to be searched: ");

                scanf("%d", &data);

                struct Node\* searchResult = searchNode(root, data);

                if (searchResult != NULL) {

                    printf("Node with data %d found.\n", data);

                } else {

                    printf("Node with data %d not found.\n", data);

                }

                break;

            case 4:

                inorderTraversal(root);

                break;

            case 5:

                preorderTraversal(root);

                break;

            case 6:

                postorderTraversal(root);

                break;

            default:

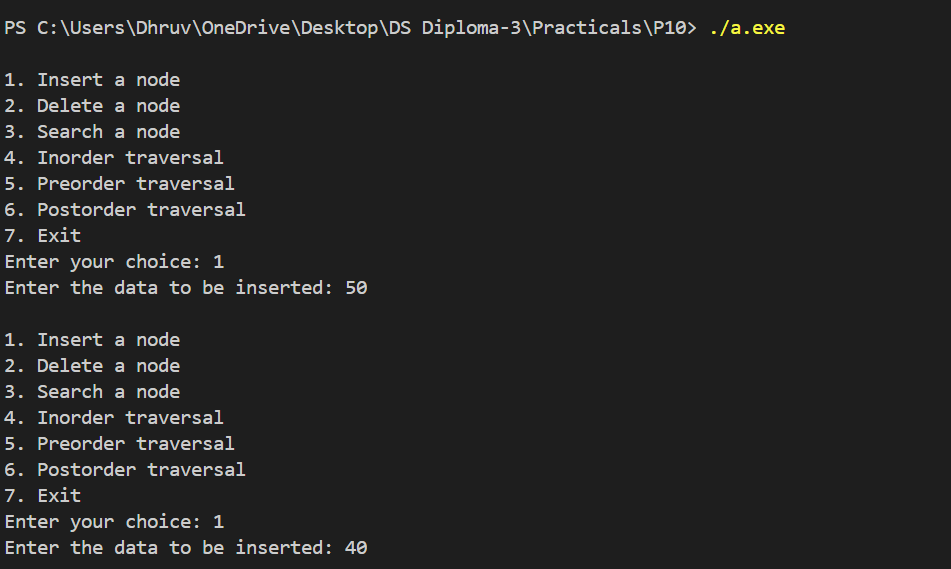
                printf("Invalid choice\n");

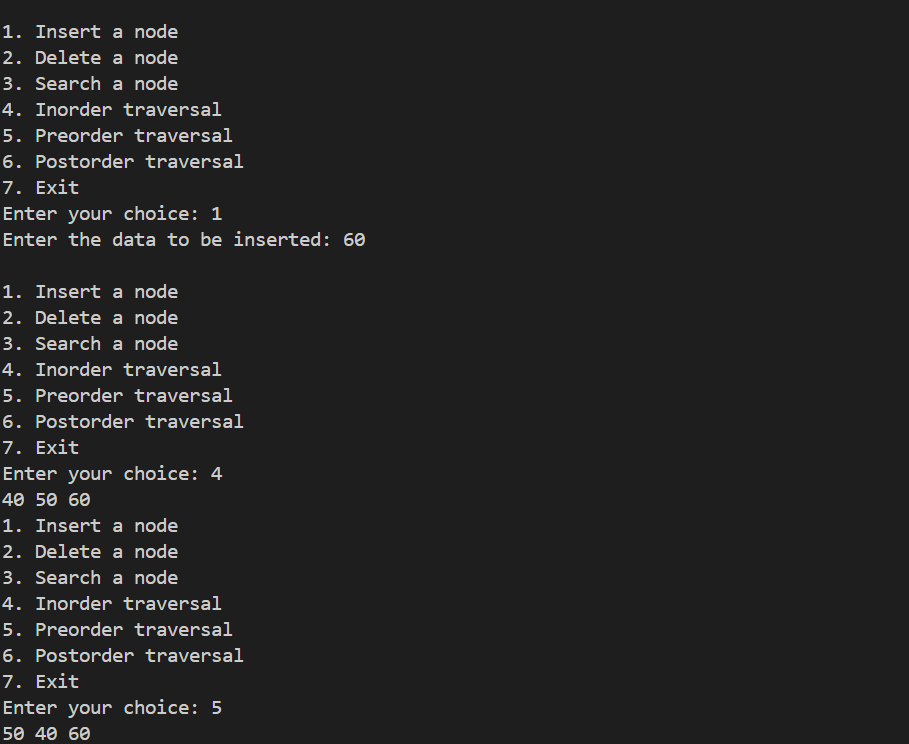
        }

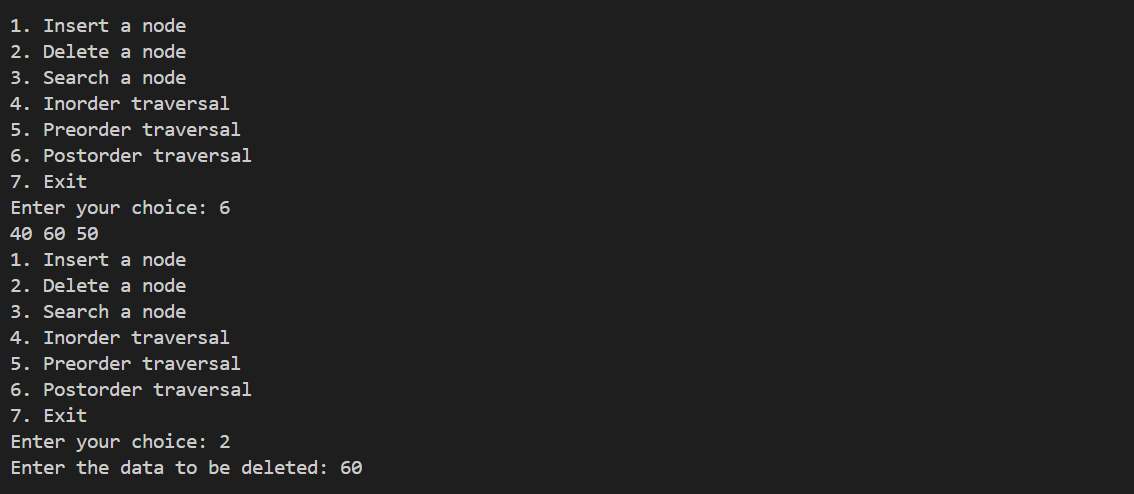
    } while (choice != 7);

}

Output :-







Practical -11

1. Write a program to implement a Bubble using Array. (A)

//1. Write a program to implement a Bubble using Array. (A)

#include <stdio.h>

#include <conio.h>

void BubbleSort(int a[],int n){

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

    {

       for(int j=0;j<n-i;j++){

        if(a[j]>a[j+1]){

            int temp=a[j];

            a[j]=a[j+1];

            a[j+1]=temp;

        }

       }

    }

}

void main()

{

    int n;

    printf("Enter Size of an array :");

    scanf("%d",&n);

    int a[n];

    for(int i=0;i<n;i++)

    {

        printf("Enter Element no %d :",(i+1));

        scanf("%d",&a[i]);

    }

    BubbleSort(a,n);

    for(int i=0;i<n;i++)

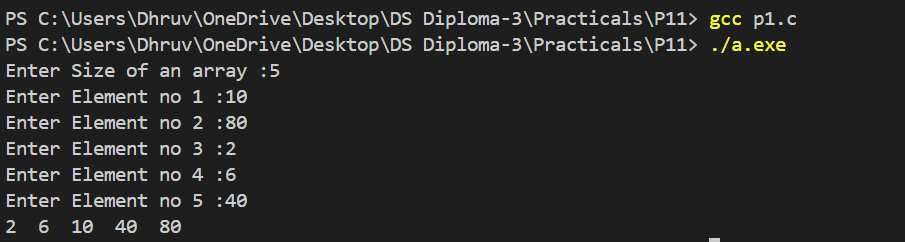
    {

        printf("%d  ",a[i]);

    }

}

Output :-



2. Write a program to implement Selection Sort using Array. (A)

//2. Write a program to implement Selection Sort using Array. (A)

#include <stdio.h>

#include <conio.h>

void SelectionSort(int a[],int n,int min){

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

    {

        min=i;

        for(int j=i+1;j<n-1;j++)

        {

            if(a[min]>a[j])

            {

                min=j;

            }

        }

        if(a[min]!=a[i])

        {

            int temp=a[i];

            a[i]=a[min];

            a[min]=temp;

        }

    }

}

void main()

{

    int n;

    printf("Enter Size of an array :");

    scanf("%d",&n);

    int a[n];

    int min;

    for(int i=0;i<n;i++)

    {

        printf("Enter Element no %d :",(i+1));

        scanf("%d",&a[i]);

    }

    SelectionSort(a,n,min);

    for(int i=0;i<n;i++)

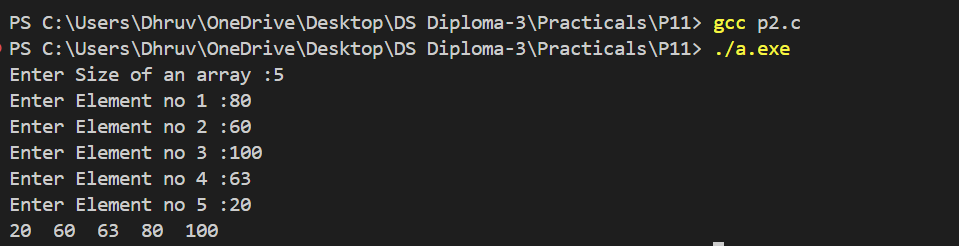
    {

        printf("%d  ",a[i]);

    }

}

Output :-



3. Write a program to implement Insertion Sort using Array. (A)

#include <stdio.h>

#include <conio.h>

void InsertionSort(int a[],int n){

for(int i=1;i<n;i++){

        int temp=a[i];

        int j=i-1;

        while(j>=0&&a[j]>temp){

            a[j+1]=a[j];

            j--;

        }

        a[j+1]=temp;

    }

}

void main()

{

    int n;

    printf("Enter Size of an array :");

    scanf("%d",&n);

    int a[n];

    for(int i=0;i<n;i++)

    {

        printf("Enter Element no %d :",(i+1));

        scanf("%d",&a[i]);

    }

    InsertionSort(a,n);

    for(int i=0;i<n;i++)

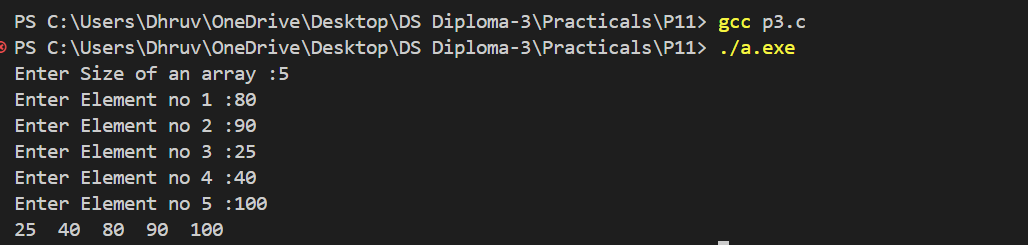
    {

        printf("%d  ",a[i]);

    }

}

Output :-



Practical -12

1. Write a program to implement a Quick Sort using Array. (A)

#include<stdio.h>

#include<conio.h>

void quicksort(int array[], int LB, int UB) {

    if (LB < UB) {

        int pivotIndex = partition(array, LB, UB);

        quicksort(array, LB, pivotIndex - 1);

        quicksort(array, pivotIndex + 1, UB);

    }

}

int partition(int array[], int LB, int UB) {

    int pivot = array[LB];

    int i = LB + 1;

    int j = UB;

    int temp;

    while (i <= j) {

        while (i <= UB && array[i] <= pivot) {

            i++;

        }

        while (j >= LB && array[j] > pivot) {

            j--;

        }

        if (i < j) {

            // Interchange value at index i and j

            temp = array[i];

            array[i] = array[j];

            array[j] = temp;

        }

    }

    // Interchange Pivot and value at jth position

    temp = array[LB];

    array[LB] = array[j];

    array[j] = temp;

    return j;

}

void main()

{

    int n;

    printf("Enter Size of an array :");

    scanf("%d",&n);

    int a[n];

    for(int i=0;i<n;i++)

    {

        printf("Enter Element no %d :",(i+1));

        scanf("%d",&a[i]);

    }

    quicksort(a,0,n-1);

    for(int i=0;i<n;i++)

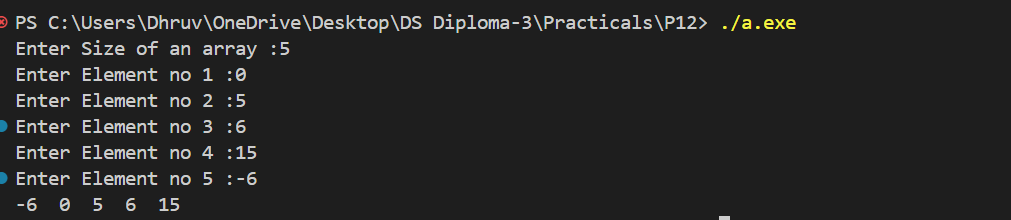
    {

        printf("%d  ",a[i]);

    }

}

Output :-



2. Write a program to implement a Merge Sort using Array. (B)

#include<stdio.h>

void merge(int a[],int len,int lb,int mid,int ub){

    int size =(ub-lb)+1;

    int b[size];

    int i=lb;

    int j=mid+1;

    int k=0;

    while(i<=mid && j<=ub){

        if(a[i]<a[j]){

            b[k]=a[i];

            i++;

        }

        else{

            b[k]=a[j];

            j++;

        }

        k++;

    }

    if(i>mid)

        while(j<=ub){

            b[k]=a[j];

            k++;

            j++;

        }

    else

        while(i<=mid){

            b[k]=a[i];

            k++;

            i++;

        }

    for(i=0;i<size;i++,lb++)

        a[lb]=b[i];

}

void mergeSort(int a[],int len,int lb,int ub){

    if(lb<ub){

        int mid =(lb+ub)/2;

        mergeSort(a,len,lb,mid);

        mergeSort(a,len,mid+1,ub);

        merge(a,len,lb,mid,ub);

    }

}

void main(){

    int i,size;

    printf("Enter Array Size :");

    scanf("%d",&size);

    int a[size];

    for(i=0;i<size;i++){

        printf("Enter Element no %d: ",(i+1));

        scanf("%d",&a[i]);

    }

    mergeSort(a,size,0,size-1);

    printf("\nSorted Array:");

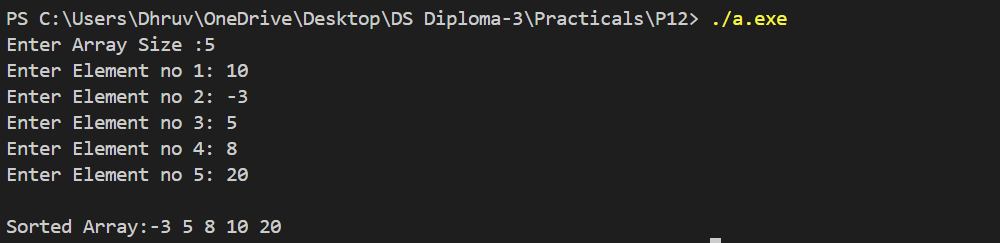
    for(i=0;i<size;i++){

        printf("%d ",a[i]);

    }

}

Output :-



Practical -13

1. Write a program to implement a Linear Search using Array. (A)

#include<stdio.h>

int linearSearch(int a[],int len,int target){

    int i;

    for(i=0;i<len;i++)

        if(a[i]== target)

            return i;

    return -1;

}

void main(){

    int i,index,target,size;

    printf("Enter Array Size :");

    scanf("%d",&size);

    int a[size];

    for(i=0;i<size;i++){

            printf("Enter Element no %d : ",(i+1));

            scanf("%d",&a[i]);

    }

    printf("Enter Search Number:");

    scanf("%d",&target);

    index = linearSearch(a,size,target);

    if(index != -1)

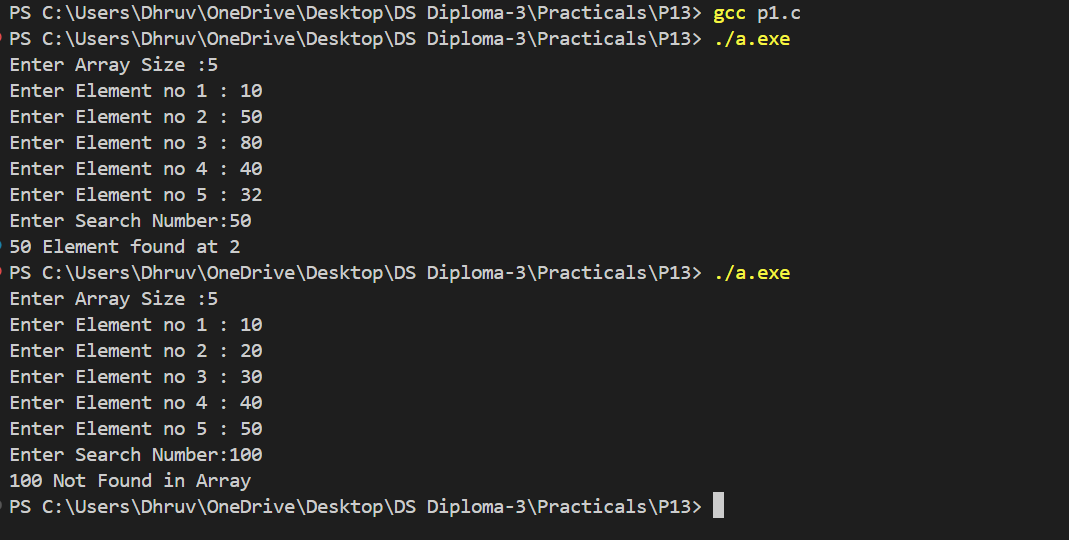
          printf("%d Element found at %d ",target,(index+1));

    else

        printf("%d Not Found in Array",target);

}

Output :-



2. Write a program to implement a binary search using Array. (A)

#include<stdio.h>

void InsertionSort(int a[],int n){

for(int i=1;i<n;i++){

        int temp=a[i];

        int j=i-1;

        while(j>=0&&a[j]>temp){

            a[j+1]=a[j];

            j--;

        }

        a[j+1]=temp;

    }

}

int binarySearch(int a[],int lb,int ub,int target){

    int mid =0;

    while(lb<=ub){

        mid = (lb+ub)/2;

        if(a[mid]==target)

            return mid;

        else if(a[mid]>target)

            ub = mid-1;

        else

            lb = mid+1;

    }

    return -1;

}

void main(){

    int i,index,target,size;

    printf("Enter Array Size :");

    scanf("%d",&size);

    int a[size];

    for(i=0;i<size;i++){

        printf("Enter Element no %d : ",(i+1));

        scanf("%d",&a[i]);

    }

    InsertionSort(a,size);

    printf("Enter Search Number:");

    scanf("%d",&target);

    printf("After Sorting \n");

    for(i=0;i<size;i++){

        printf("Index %d -> %d \n ",(i+1),a[i]);

    }

    index = binarySearch(a,0,size,target);

    if(index != -1)

        printf("%d Element found at %d ",target,(index+1));

    else

        printf("%d Not Found in Array",target);

}

Output :-

