

**DATA STRUCTURE AND C LABORATORY**

# **NAME- HRISHA GOSSAIN**

# **COURSE- MCA**

# **STREAM- MCA**

# **SEMESTER- 1**

# **SECTION- B**

# **ClASS ROLL- 55**

# **ENROLLMENT NO.- 12023006015068**

# **REGISTRATION NO.-**

**Index**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SI. No.** | **Assignment** | **Page No.** | **Signature** | **Grade** |
| 1. | Assignment 1 | 2 - 15 |  |  |
| 2. | Assignment 2 | 16 - 25 |  |  |
| 3. | Assignment 3 | 26 - 44 |  |  |
| 4. | Assignment 4 | 45 - 57 |  |  |
| 5. | Assignment 5 | 58 - 66 |  |  |
| 6. | Assignment 6 | 67 - 85 |  |  |

**ASSIGNMENT 1**

**Question 1:** Write a C program to print an array.

**Source Code :**

#include<stdio.h>

int main(){

int i,n;

printf("Enter The Number Of Terms : ");

scanf("%d",&n);

int a[n];

printf("\nEnter The Element Of The Array : ");

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

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

}

printf("\nentered Array Is : ");

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

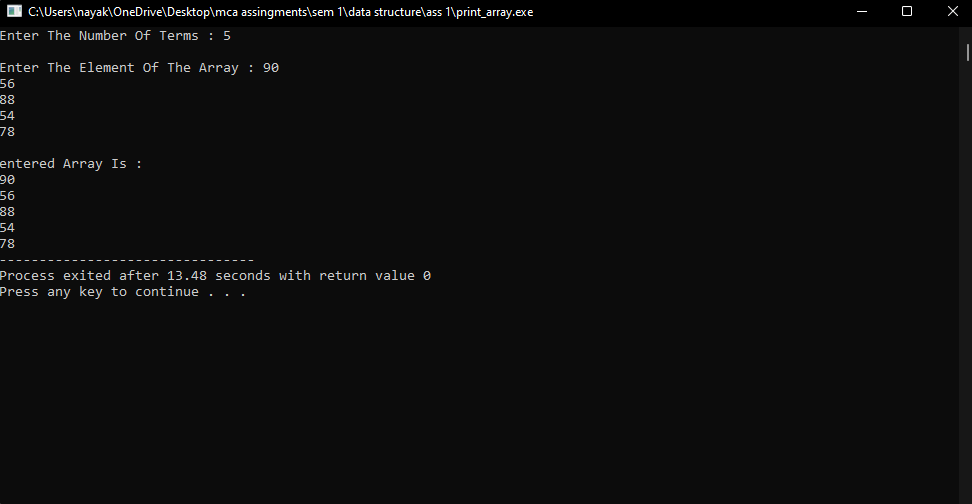
printf("\n%d",a[i]);

}

return 0;

}

**Output :**

****

**Question 2:** write a program to check whether the given string is palindrom or not.

**Source Code :**

#include<stdio.h>

int main()

{

char string[100];

int length=0, flag=1,i;

printf("Enter string:\n");

gets(string);

for(i=0;string[i]!='\0';i++)

{

length++;

}

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

{

if( string[i] != string[length-1-i] )

{

flag=0;

break;

}

}

if(flag==1)

{

printf("The String Is Palindrom");

}

else

{

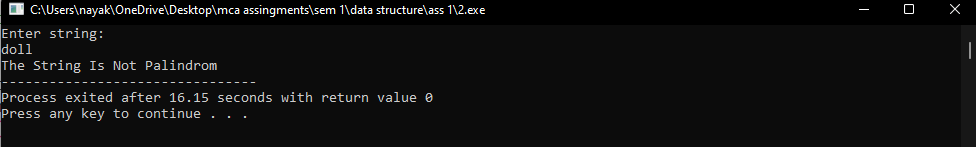
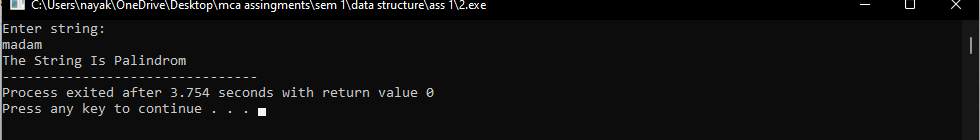
printf("The String Is Not Palindrom");

}

return 0;

}

**Output :**

****

**Question 3:** Write a C program to convert temperature from degree Centigrade to Fahrenheit.

**Source Code :**

#include <stdio.h>

int main()

{

float celsius, fahrenheit;

printf("Enter temperature in celsius: ");

scanf("%f", &celsius);

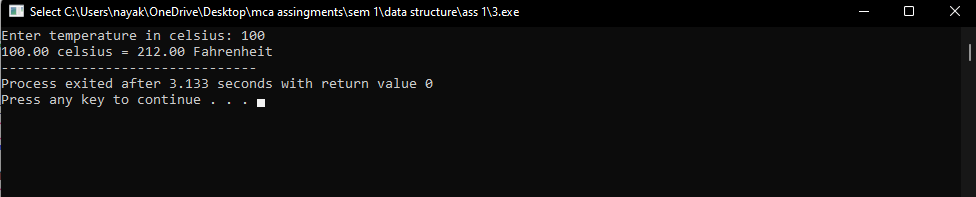
fahrenheit = (celsius \* 9 / 5) + 32;

printf("%.2f celsius = %.2f Fahrenheit", celsius, fahrenheit);

return 0;

}

**Output :**

****

**Question 4:** Write a C program to sort an array.

**Source Code :**

#include<stdio.h>

int main(){

int i,n,j,swap;

printf("Enter The Number Of Terms : ");

scanf("%d",&n);

int a[n];

printf("\nEnter The Element Of The Array : ");

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

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

}

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

for(j=0;j<n;j++){

if(a[i]<a[j]){

swap=a[i];

a[i]=a[j];

a[j]=swap;

}

}

}

printf("\nThe Sorted Array Is : ");

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

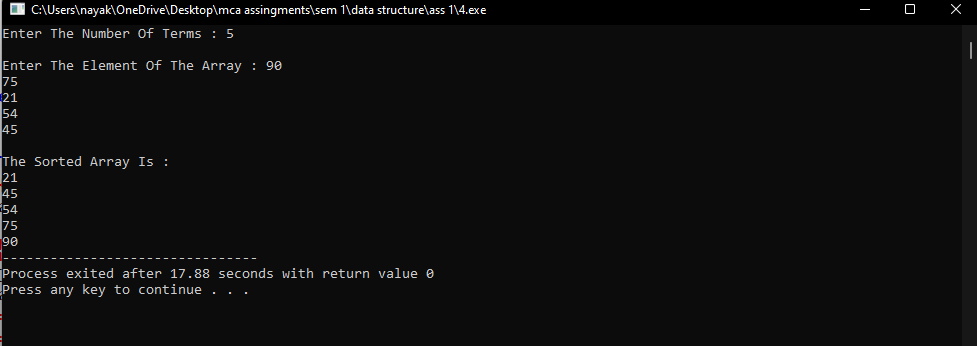
printf("\n%d",a[i]);

}

return 0;

}

**Output :**

****

**Question 5:** Write a C program to print the largest and second largest element of the array.

**Source Code :**

#include<stdio.h>

int main(){

int i,n,max=0,max\_index,smax=0,smax\_index;

printf("Enter The Number Of Terms : ");

scanf("%d",&n);

int a[n];

printf("\nEnter The Element Of The Array : ");

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

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

}

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

if(a[i]>max){

max=a[i];

max\_index=i;

}

}

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

if(a[i]>smax && max\_index!=i){

smax=a[i];

smax\_index=i;

}

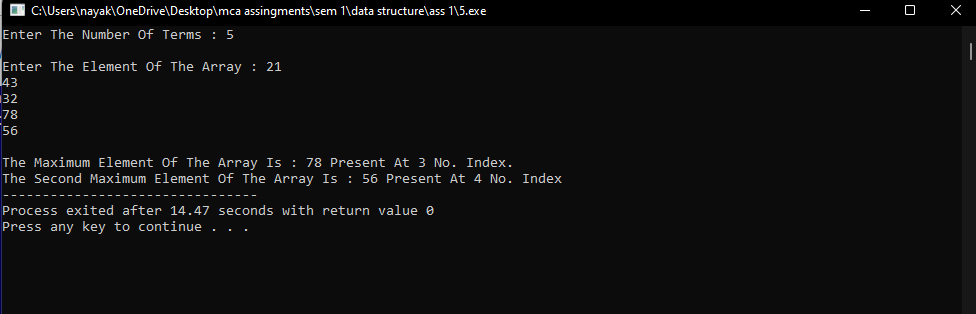
}

printf("\nThe Maximum Element Of The Array Is : %d Present At %d No. Index. \nThe Second Maximum Element Of The Array Is : %d Present At %d No. Index",max,max\_index,smax,smax\_index);

return 0;

}

**Output :**



**Question 6:** Write a C program to display Fibonacci series.

**Source Code :**

#include<stdio.h>

int Fibonacci(int);

int main()

{

int n, i, j=0;

printf("Enter the range : ");

scanf("%d",&n);

printf("Fibonacci series :\n");

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

{

printf("%d ", Fibonacci(i));

j++;

}

return 0;

}

int Fibonacci(int n)

{

if ( n == 0 )

return 0;

else if ( n == 1 )

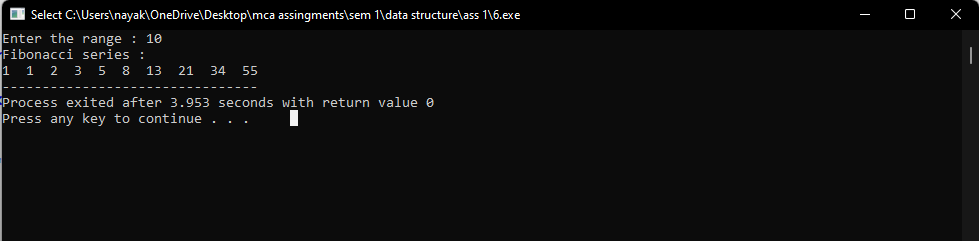
return 1;

else

return ( Fibonacci(n-1) + Fibonacci(n-2) );

}

**Output :**

****

**Question 7:** Write a C program that reads two 2D matrices from console verifies if matrics multiplication is possible or not and then multiply matrices and print the 3rd matrix.

**Source Code :**

#include<stdio.h>

int main(){

int r1, c1, r2, c2,i,j,k;

printf("Enter rows and column for the first matrix: ");

scanf("%d %d", &r1, &c1);

printf("Enter rows and column for the second matrix: ");

scanf("%d %d", &r2, &c2);

while (c1 != r2) {

printf("Error! Enter rows and columns again.\n");

printf("Enter rows and columns for the first matrix: ");

scanf("%d%d", &r1, &c1);

printf("Enter rows and columns for the second matrix: ");

scanf("%d%d", &r2, &c2);

}

int first[r1][c1], second[r2][c2], result[r1][c1];

printf("\nEnter values for the first matrix:\n ");

for(i=0;i<r1;i++){

for(j=0;j<c1;j++){

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

}

}

printf("\nEnter values for the second matrix:\n ");

for(i=0;i<r2;i++){

for(j=0;j<c2;j++){

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

}

}

for(i=0;i<r1;i++){

for(j=0;j<c1;j++){

result[i][j]=0;

for(k=0;k<r1;k++){

result[i][j]+=first[i][k]\*second[k][i];

}

}

}

printf("\nThe resulting matrix is:\n ");

for(i=0;i<r1;i++){

for(j=0;j<c1;j++){

printf("%d ",result[i][j]);

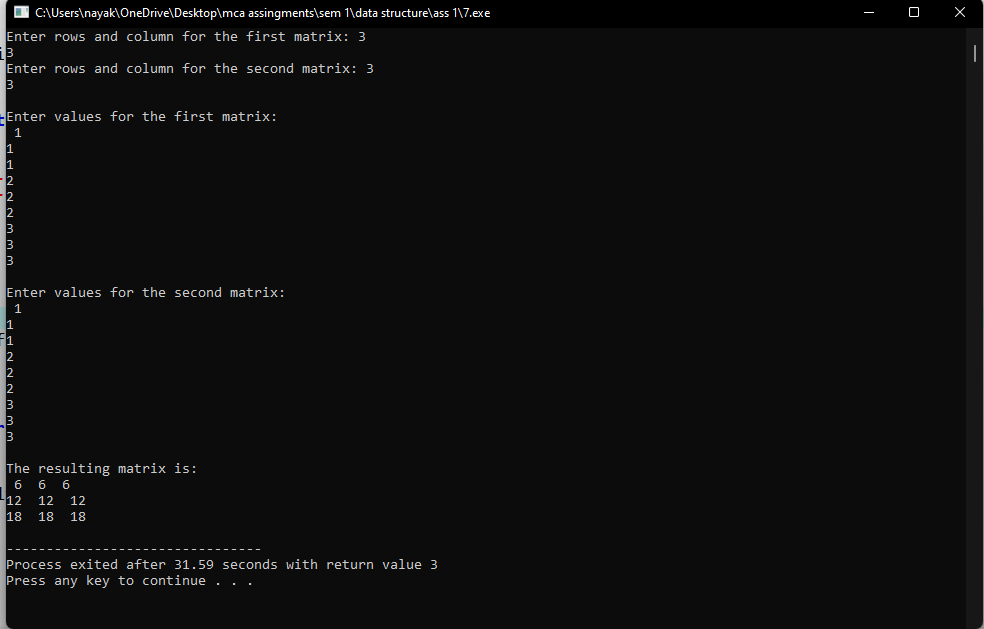
}

printf("\n");

}

}

**Output :**

****

**Question 8:** Write a program that reads a 2D metrics and checks if the metrics is a symmetric metrics or not.

**Source Code :**

#include<stdio.h>

int main()

{

int i, j, r,c, a[10][10], b[10][10], count = 1;

printf("\n Please enter number of rows and column : ");

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

printf("\n Please Enter the matrix elements \n");

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

{

for(j = 0;j < c;j++)

{

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

}

}

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

{

for(j = 0;j < c; j++)

{

b[j][i] = a[i][j];

}

}

printf("The transpose matrix is=\n");

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

{

for(j = 0;j < c; j++)

{

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

}

printf("\n");

}

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

{

for(j = 0; j < c; j++)

{

if(a[i][j] != b[i][j])

{

count++;

break;

}

}

}

if(count == 1)

{

printf("\n The Matrix that you entered is a symmetric Matrix ");

}

else

{

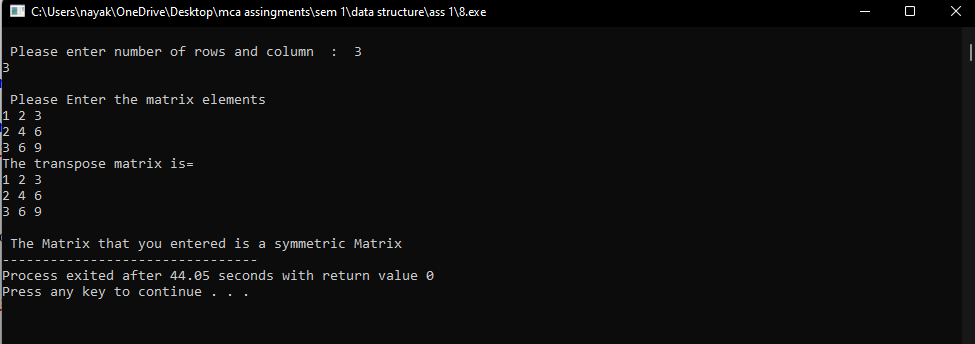
printf("\n The Matrix that you entered is Not a symmetric Matrix ");

}

return 0;

}

**Output :**

****

**Question 9:** write a program to print reverse array

**Source Code :**

#include<stdio.h>

int main(){

int i,n;

printf("Enter The Number Of Terms : ");

scanf("%d",&n);

int a[n];

printf("\nEnter The Element Of The Array : ");

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

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

}

printf("\nentered Array Is : ");

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

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

}

printf("\nreverse Array Is : ");

for(i=n-1;i>=0;i--){

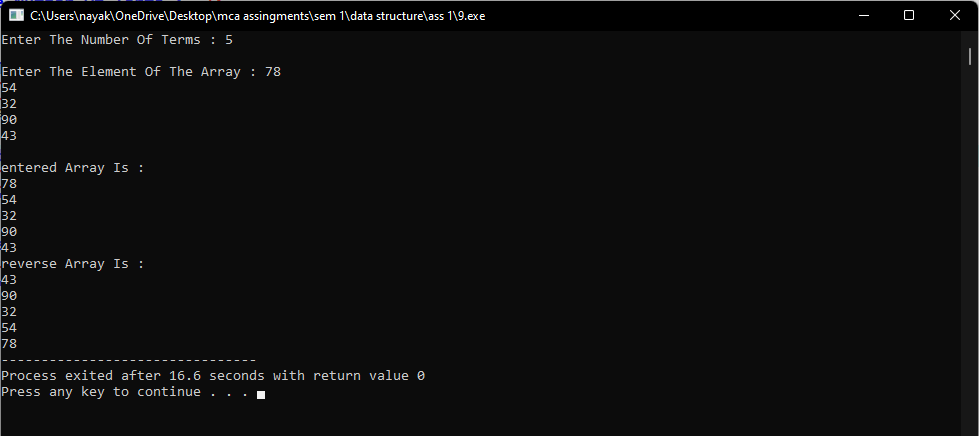
printf("\n%d",a[i]);

}

return 0;

}

**Output :**

****

**Question 10:** Write a C program to do sum of all elements of an array

**Source Code :**

#include<stdio.h>

int main(){

int i,n,sum=0;

printf("Enter The Number Of Terms : ");

scanf("%d",&n);

int a[n];

printf("\nEnter The Element Of The Array : ");

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

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

}

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

sum=sum+a[i];

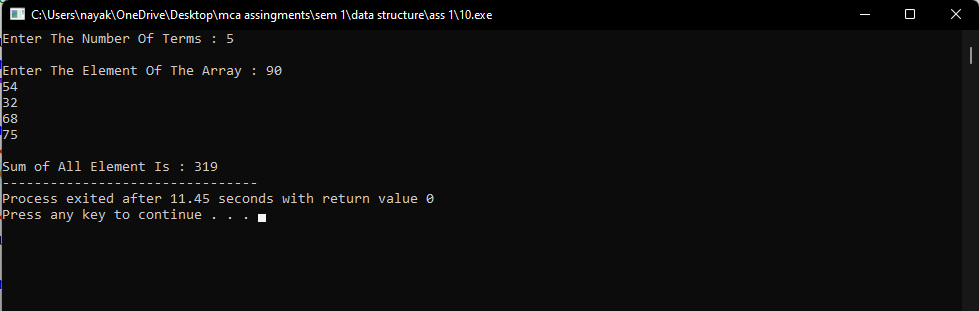
}

printf("\nSum of All Element Is : %d",sum);

return 0;

}

**Output :**



**Question 11:** Write a C program to check duplicate number in an array.

**Source Code :**

#include<stdio.h>

int main(){

int i,j,n,search,count;

printf("Enter The Number Of Terms : ");

scanf("%d",&n);

int a[n];

printf("\nEnter The Element Of The Array : ");

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

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

}

printf("\nentered Array Is : ");

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

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

}

printf("\n");

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

search= a[i];

count=0;

for(j=0;j<n;j++){

if(i!=j && search==a[j]){

count++;

}

}

if(count==0){

printf("\na[%d]=%d ---> No duplicate present in array",i,search);

}

else{

printf("\na[%d]=%d ---> Present %d more time in the array",i,search,count);

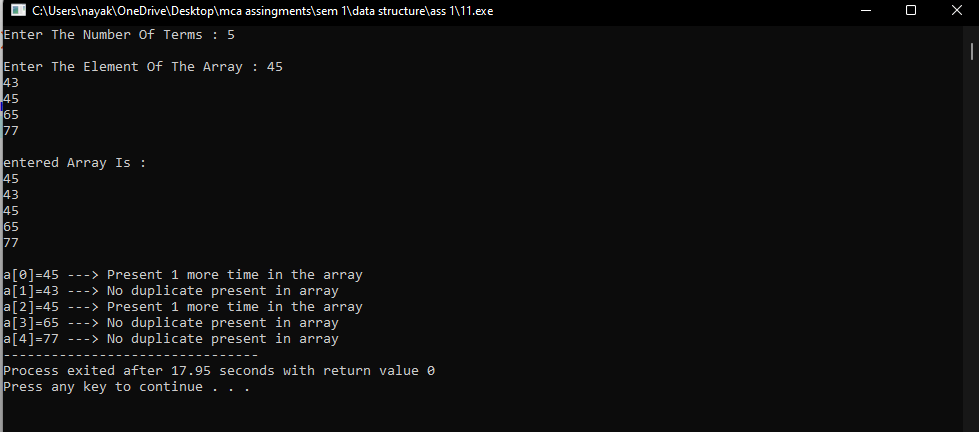
}

}

return 0;

}

**Output :**

****

**ASSIGNMENT 2**

**Question 1:** Write a C program to read a 2D array (with most of the elements as 0s) and then represent the same array as Sparse Metrics.

**Source Code :**

#include <stdio.h>

#define MAX\_ROWS 100

#define MAX\_COLS 100

void convertToSparseMatrix(int matrix[MAX\_ROWS][MAX\_COLS], int rows, int cols) {

int sparseMatrix[3][MAX\_ROWS \* MAX\_COLS];

int k = 0; // Counter for non-zero elements in the sparse matrix

int i,j;

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

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

if (matrix[i][j] != 0) {

sparseMatrix[0][k] = i; // Row index

sparseMatrix[1][k] = j; // Column index

sparseMatrix[2][k] = matrix[i][j]; // Value

k++;

}

}

}

printf("\nSparse Matrix Representation:\n");// Print the sparse matrix

printf("Row\nColumn\nValue\n");

for (i = 0; i < 3; ++i) {

for (j = 0; j < k; ++j) {

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

}

printf("\n");

}

}

int main() {

int rows, cols, i, j;

printf("Enter the number of rows and columns of the matrix: ");

scanf("%d %d", &rows, &cols);

if (rows <= 0 || cols <= 0 || rows > MAX\_ROWS || cols > MAX\_COLS) {

printf("Invalid input for rows or columns.\n");

return 1;

}

int matrix[MAX\_ROWS][MAX\_COLS];

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

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

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

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

}

}

printf("Elements of the 2d array : \n");

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

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

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

}

printf("\n");

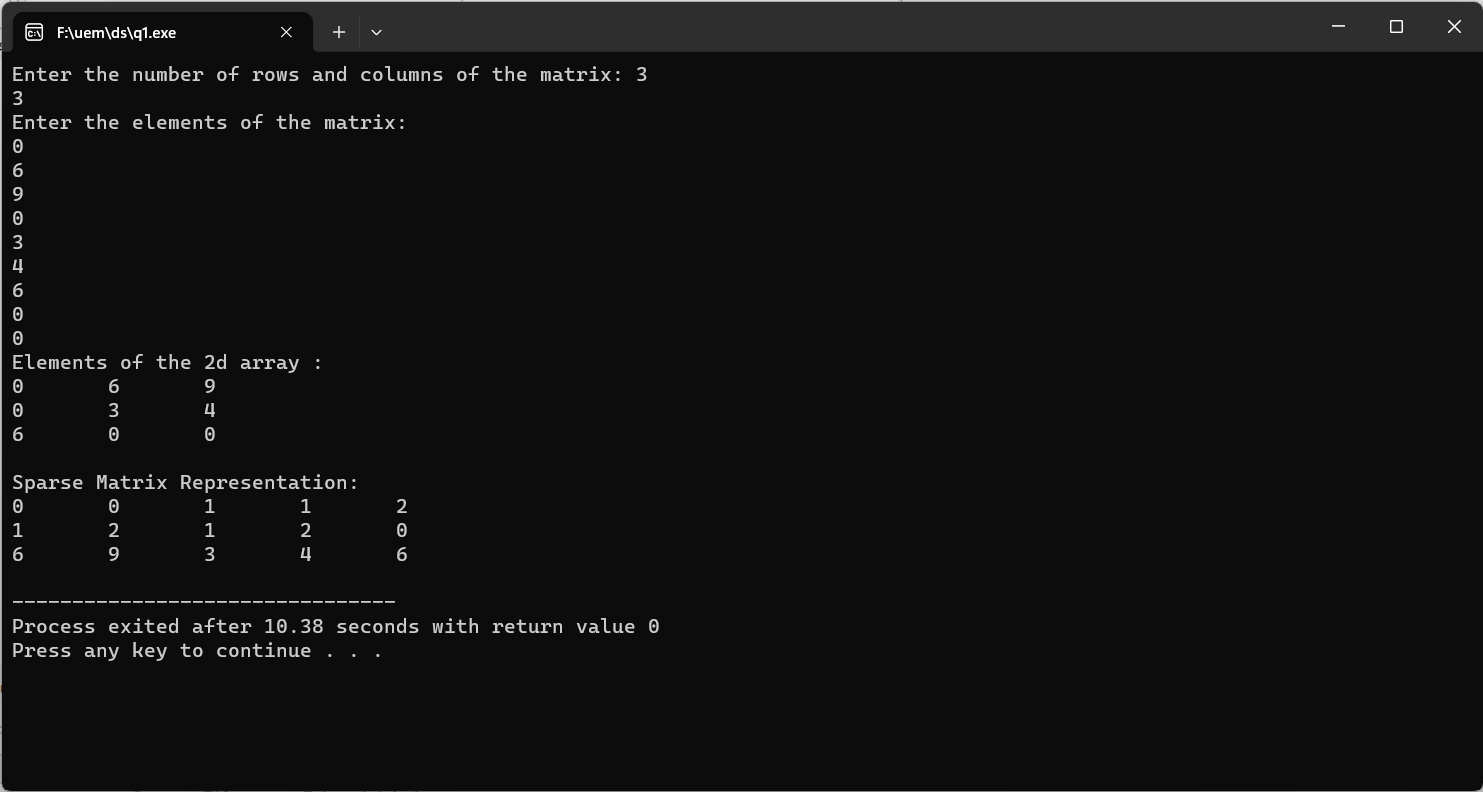
}

convertToSparseMatrix(matrix, rows, cols); }// Convert and print the sparse matrix representation

return 0;

}

**Output:**



**Question 2:** Write a C program to pass an array to a function using Call by Value, update the array values in the function, print the array elements both in the function and in the calling function.

**Source Code:**

#include <stdio.h>

void updateArray(int arr[], int size) {

int i;

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

arr[i] += 10;

}

printf("Array elements inside the function:\n");// Print array elements inside the function

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

printf("%d ", arr[i]);

}

printf("\n");

}

void main() {

int i;

int myArray[] = {1, 2, 3, 4, 5};

int size = sizeof(myArray) / sizeof(myArray[0]);

printf("Array elements before calling the function:\n");//Print array before calling the function

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

printf("%d ", myArray[i]);

}

printf("\n");

updateArray(myArray, size);// Call the function and pass the array

printf("Array elements after calling the function:\n");// Print array after calling the function

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

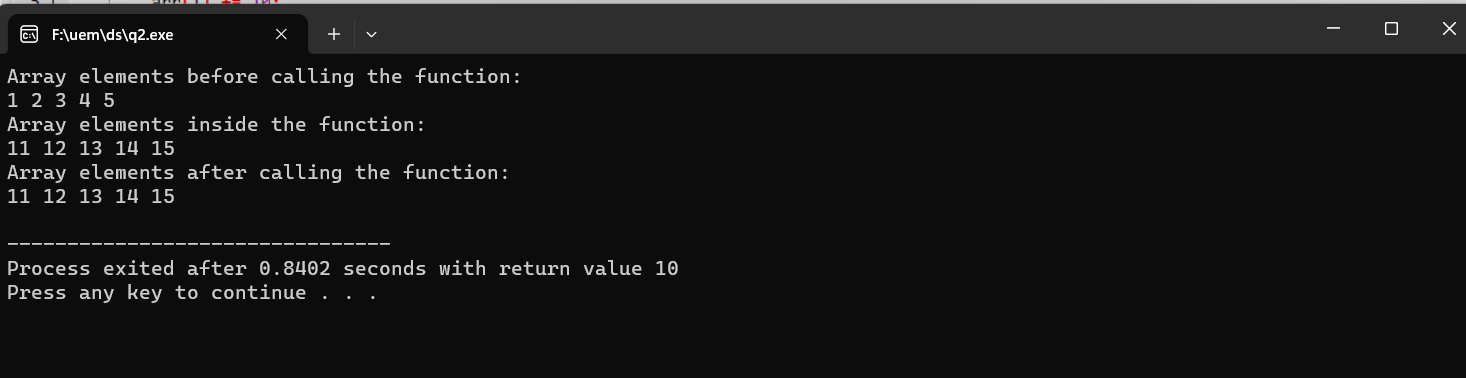
printf("%d ", myArray[i]);

}

printf("\n");

}

**Output:**



**Question 3:** Write a C program to pass an array to a function using Call by Reference, update the array values in the function, print the array elements both in the function and in the calling function.

**Source Code:**

#include <stdio.h>

void updateArray(int \*arr, int size) {

int i;

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

arr[i] += 10;

}

printf("Array elements inside the function:\n");// Print array elements inside the function

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

printf("%d ", arr[i]);

}

printf("\n");

}

void main() {

int i;

int myArray[] = {10,9,8,7,6,5};

int size = sizeof(myArray) / sizeof(myArray[0]);

printf("Array elements before calling the function:\n");// Print array elements before calling the function

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

printf("%d ", myArray[i]);

}

printf("\n");

updateArray(myArray, size);// Call the function and pass the array by reference

printf("Array elements after calling the function:\n"); // Print array elements after calling the function

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

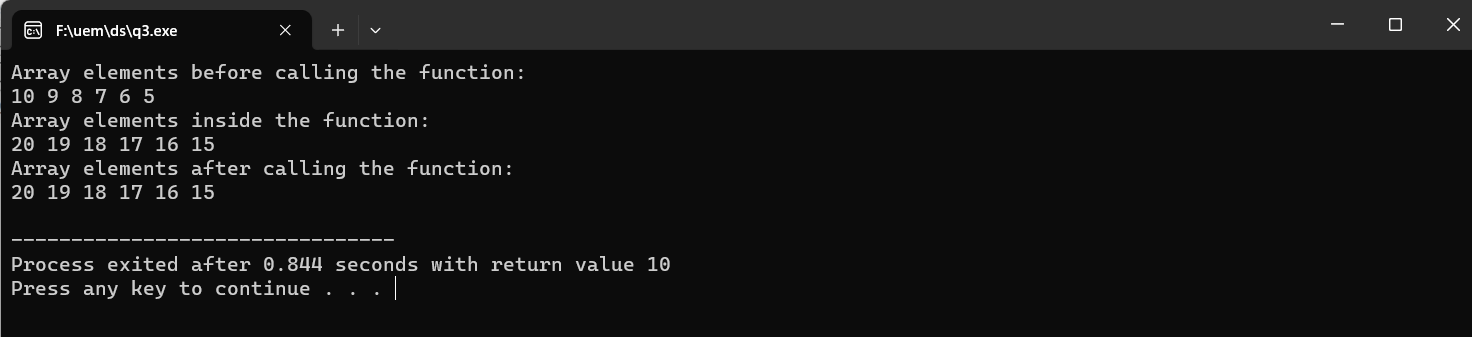
printf("%d ", myArray[i]);

}

printf("\n");

}

**Output:**



**Question 4:** Write a program to display n number of elements. Memory should be allocated dynamically using malloc( ).

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

int main() {

int n,i;

printf("Enter the number of elements: ");// Get the number of elements from the user

scanf("%d", &n);

int \*arr = (int \*)malloc(n \* sizeof(int));// Dynamically allocate memory for the array

if (arr == NULL) {// Check if memory allocation is successful

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

return 1; // Exit with an error code

}

printf("Enter %d elements:\n", n);// Input elements from the user

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

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

}

printf("Entered elements are:\n");

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

printf("%d ", arr[i]);

}

printf("\n");

free(arr); // Free the dynamically allocated memory

return 0;

}

**Output:**



**Question 5:**  Write a program to display n number of elements. Memory should be allocated dynamically using calloc( ).

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

void main() {

int i, n;

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

scanf("%d", &n);

int \*arr = (int \*)calloc(n, sizeof(int));// Dynamically allocate memory for the array using calloc

if (arr == NULL) {// Check if memory allocation is successful

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

return 1; // Exit with an error code

}

printf("Enter %d elements:\n", n);

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

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

}

printf("Entered elements are:\n");

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

printf("%d ", arr[i]);

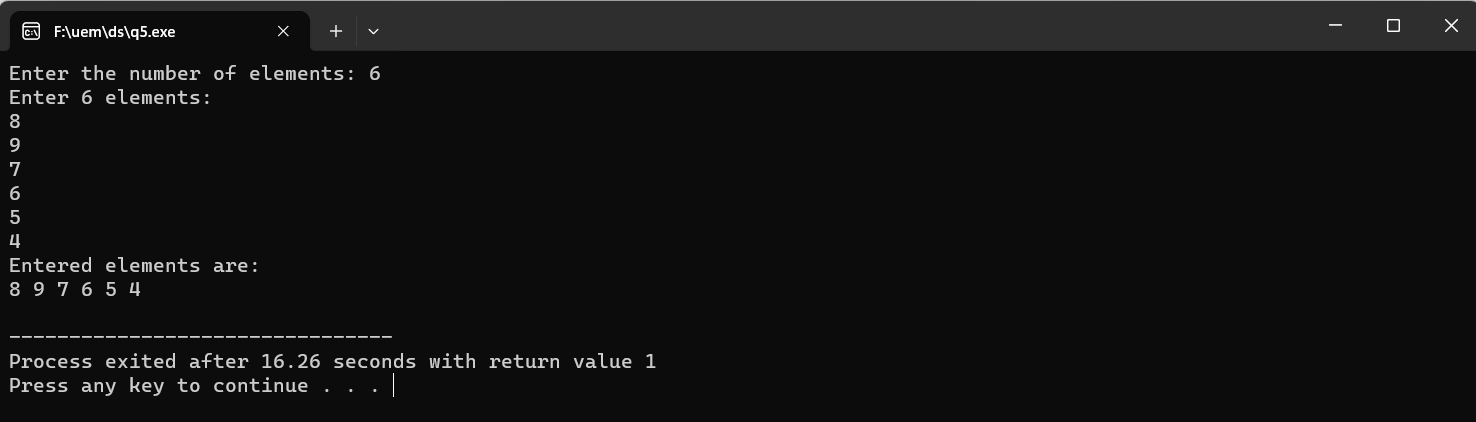
}

printf("\n");

free(arr);// Free the dynamically allocated memory

}

**Output:**



**Question 6:** Write a program to allocate memory using malloc( ) and then reallocate the previously  allocated memory using realloc( ). Display the elements which have been taken after reallocation.

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

int main() {

int i, n;

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

scanf("%d", &n);

int \*arr = (int \*)malloc(n \* sizeof(int)); // Dynamically allocate memory for the initial array using malloc

if (arr == NULL) {// Check if memory allocation is successful

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

return 1; // Exit with an error code

}

printf("Enter %d elements:\n", n);

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

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

}

printf("Elements before reallocation:\n");// Display the elements before reallocation

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

printf("%d ", arr[i]);

}

printf("\n");

int newSize;// Get the new size from the user for reallocation

printf("Enter the new size for reallocation: ");

scanf("%d", &newSize);

arr = (int \*)realloc(arr, newSize \* sizeof(int)); // Reallocate the memory using realloc

if (arr == NULL) {// Check if reallocation is successful

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

return 1; // Exit with an error code

}

printf("Enter %d elements after reallocation:\n", newSize);

for (i = 0; i < newSize; i++) {

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

}

printf("Elements after reallocation:\n");

for (i = 0; i < newSize; i++) {

printf("%d ", arr[i]);

}

printf("\n");

free(arr);// Free the dynamically allocated memory

return 0;

}

**Output:**



**Question 7:** Write a program to allocate memory using calloc( ) and then reallocate the previously  allocated memory using realloc( ). Display the elements which have been taken after reallocation.

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

int main() {

int i,n;

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

scanf("%d", &n);

int \*arr = (int \*)calloc(n, sizeof(int));// Dynamically allocate memory for the initial array using calloc

if (arr == NULL) {// Check if memory allocation is successful

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

return 1; // Exit with an error code

}

printf("Enter %d elements:\n", n);

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

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

}

printf("Elements before reallocation:\n"); // Display the elements before reallocation

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

printf("%d ", arr[i]);

}

printf("\n");

int newSize;// Get the new size from the user for reallocation

printf("Enter the new size for reallocation: ");

scanf("%d", &newSize);

arr = (int \*)realloc(arr, newSize \* sizeof(int));// Reallocate the memory using realloc

if (arr == NULL) {// Check if reallocation is successful

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

return 1; // Exit with an error code

}

printf("Enter %d elements after reallocation:\n", newSize);// Input elements after reallocation

for (i = 0; i < newSize; i++) {

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

}

printf("Elements after reallocation:\n");// Display the elements after reallocation

for (i = 0; i < newSize; i++) {

printf("%d ", arr[i]);

}

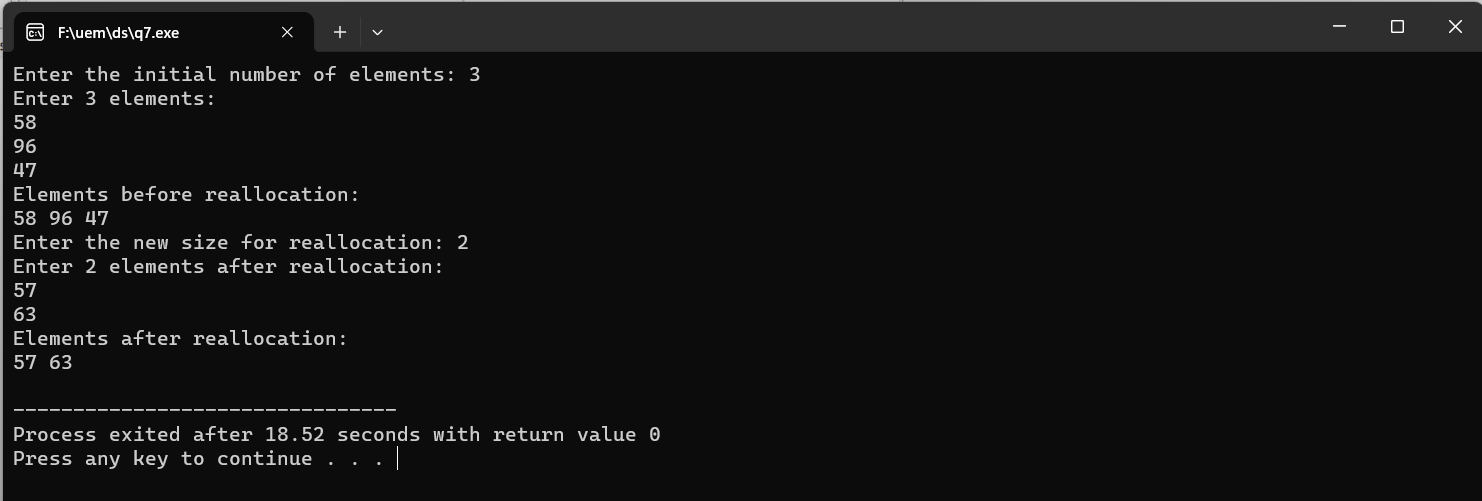
printf("\n");

free(arr);// Free the dynamically allocated memory

return 0;

}

**Output:**



**Question 8:** Write a C program to search an element in an Array using dynamic memory allocation.

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

int searchElement(int \*arr, int size, int target) {

int i;

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

if (arr[i] == target) {

return i; // Return the index if the element is found

}

}

return -1; // Return -1 if the element is not found

}

int main() {

int i,n;

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

scanf("%d", &n);

int \*arr = (int \*)malloc(n \* sizeof(int));// Dynamically allocate memory for the array

if (arr == NULL) {// Check if memory allocation is successful

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

return 1; // Exit with an error code

}

printf("Enter %d elements:\n", n);

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

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

}

int target;

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

scanf("%d", &target);

int index = searchElement(arr, n, target);// Search for the element in the array

if (index != -1) {// Display the result

printf("Element %d found at index %d.\n", target, index);

} else {

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

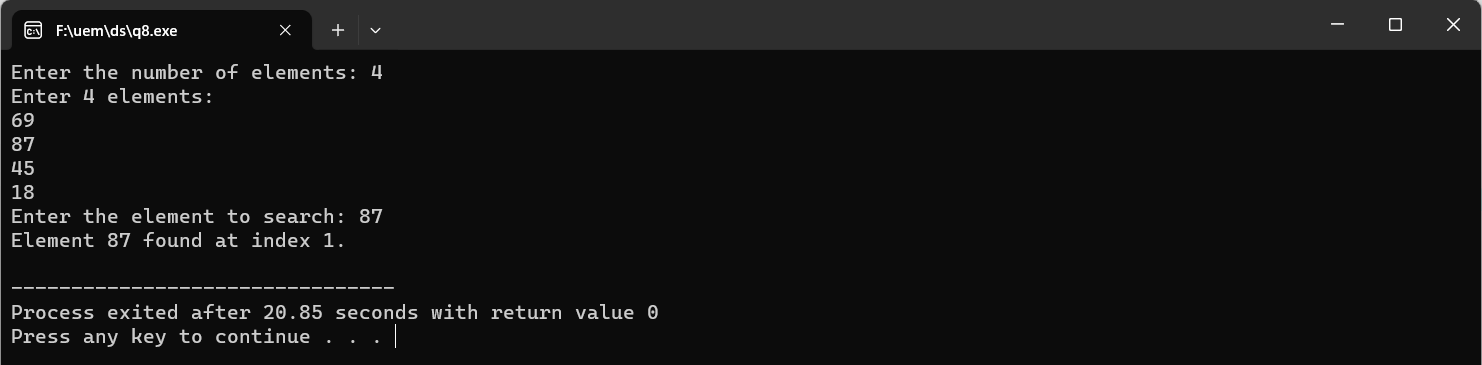
}

free(arr);// Free the dynamically allocated memory

return 0;

}

**Output:**



**ASSIGNMENT 3**

**Question 1:**  Write a Menu driven C program to accomplish the following functionalities in single linked list.

a) Create a single linked list.

b) Display the elements of a single linked list.

c) Insert a node at the beginning of a single linked list.

d) Insert a node at the end of a single linked list.

e) Insert a node before a given node of a single linked list.

f) Insert a node after a given node of a single linked list.

g) Delete a node from the beginning of a single linked list.

h) Delete a node from the end of a single linked list.

i) Delete a node after a given node of a single linked list.

j) Delete the entire single linked list.

**Source Code :**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

int data;

struct node\* link;

};

struct node\* insert\_before\_value(struct node\* head){

if(head==NULL)

printf("The linked list is empty");

else {

printf("Enter the value before which data should be inserted:");

int odata;

scanf("%d",&odata);

if(head->data==odata){

struct node\* temp=(struct node\*)malloc(sizeof(struct node));

int item;

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

scanf("%d",&item);

temp->data=item;

temp->link=head;

head=temp;

}

else{

if(head->link==NULL) //there is only 1 node and data not found

printf("The data is not found");

else{ // there is more than one node and 1st node does not have data

struct node\* temp1=head, \*temp3,\*temp2;

temp3=temp1;

temp1=temp1->link;

while(temp1!= NULL){

if(temp1->data==odata){

struct node\* temp2=(struct node\*)malloc(sizeof(struct node));

int item;

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

scanf("%d",&item);

temp2->data=item;

temp2->link=temp1;

temp3->link=temp2;

break;

}

temp3=temp1;

temp1=temp1->link;

}

if(temp1==NULL){

printf("The data is not found");

}

}

}

}

return head;

}

struct node\* insert\_in\_beg(struct node\* head)

{

struct node\* temp;

int item;

temp=(struct node\*)malloc(sizeof(struct node));

if(temp==NULL)

printf("the node has not been created");

else

{

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

scanf("%d",&item);

temp->data=item;

temp->link=head;

head=temp;

}

return (head);

}

void display(struct node\* head)

{

struct node\* temp;

if(head==NULL)

printf("the linked list is empty");

else

{

printf("\nthe elements of the linked list are:");

temp=head->link;

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

while(temp!=NULL)

{

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

temp=temp->link;

}

printf("\n");

}

}

struct node\* insert\_at\_end(struct node\* head)

{

struct node\* temp,\*temp1;

int item;

temp=(struct node\*)malloc(sizeof(struct node));

if(temp==NULL)

printf("\nthe node has not been created");

else

{

printf("enter an item to be inserted:");

scanf("%d",&item);

if(head==NULL)

{

temp->data=item;

temp->link=NULL;

head=temp;

}

else

{

temp1=head;

while(temp1->link!=NULL)

{

temp1=temp1->link;

}

temp1->link=temp;

temp->data=item;

temp->link=NULL;

}

}

return (head);

}

struct node\* creation(struct node\* head)

{

struct node\* temp=(struct node\*)malloc(sizeof(struct node));

printf("Enter the data:");

scanf("%d",&temp->data);

temp->link=NULL;

head=temp;

return head;

}

struct node\* del\_in\_beg(struct node\* head)

{

if(head==NULL)

printf("\nThe linked list is empty");

else

{

head=head->link;

}

return head;

}

struct node \* del\_at\_end(struct node\* head)

{

struct node\* temp, \*temp1;

if(head==NULL)

printf("\nthe linked list is empty");

else if(head->link==NULL)

head=NULL;

else

{

temp=head;

temp1=temp->link;

while(temp1->link!=NULL)

{

temp=temp1;

temp1=temp1->link;

}

temp->link=NULL;

}

return head;

}

struct node\* del\_entire(struct node\* head)

{

head=NULL;

return head;

}

void insert\_after\_val(struct node\* head)

{

if(head==NULL)

{

printf("\nthe linked list is empty");

}

else

{

struct node\* temp, \* temp1, \*temp2;

int item,val;

temp=(struct node\*) malloc(sizeof(struct node));

if(temp==NULL)

printf("the node has not been created");

else

{

printf("enter the item to insert:");

scanf("%d",&item);

printf("enter the value after which %d will be inserted:",item);

scanf("%d",&val);

temp1=head;

temp2=temp1->link;

while(temp2!=NULL)

{

if(temp1->data==val)

break;

temp1=temp2;

temp2=temp2->link;

}

if(temp1->data!=val)

printf("\nthe value is not available in the linked list");

else

{

temp1->link=temp;

temp->link=temp2;

temp->data=item;

}

}

}

}

void del\_after\_val(struct node\* head)

{

if(head==NULL)

printf("the linked list is empty");

else

{

struct node\* temp1, \* temp2;

int val;

printf("enter the value after which element will be deleted");

scanf("%d",&val);

temp1=head;

temp2=temp1->link;

while(temp2!=NULL)

{

if(temp1->data==val)

break;

temp1=temp2;

temp2=temp2->link;

}

if(temp1->data!=val)

printf("the value doesn't exist");

else

{

if(temp2==NULL)

printf("no element remains to be deleted after this value");

else

{

temp1->link=temp2->link;

}

}

}

}

int main()

{

struct node\* head=NULL;

int ch;

while(1)

{

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

printf("1.Creation \n2.display \n3.insertion in the beginning");

printf("\n4.insertion at the end\n5.insertion before a value\n6.insertion after a value\n");

printf("7.deletion in the beginning \n8.deletion at the end \n9.deletion after a value\n10. Delete entire list");

printf("\n11.exit\nenter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 3: head=insert\_in\_beg(head);

break;

case 4: head=insert\_at\_end(head);

break;

case 1: head=creation(head);

break;

case 7: head=del\_in\_beg(head);

break;

case 8: head=del\_at\_end(head);

break;

case 10: head=del\_entire(head);

break;

case 9: del\_after\_val(head);

break;

case 6: insert\_after\_val(head);

break;

case 2: display(head);

break;

case 5: head=insert\_before\_value(head);

break;

case 11: exit(0);

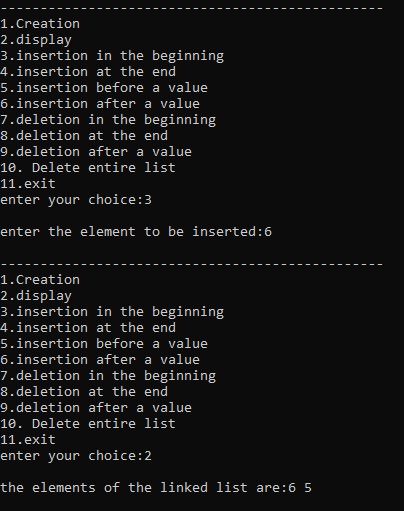
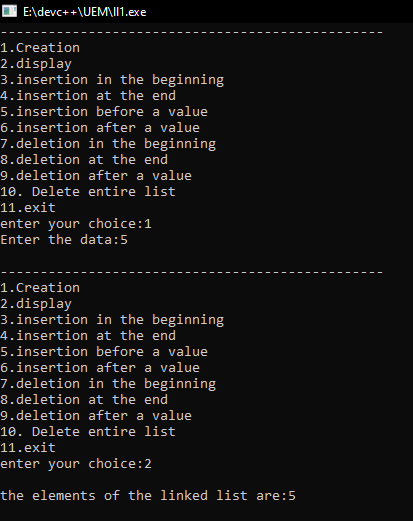
default:printf("you have entered a wrong choice");

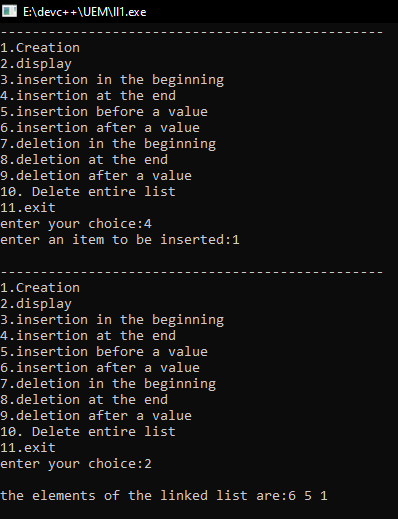
}

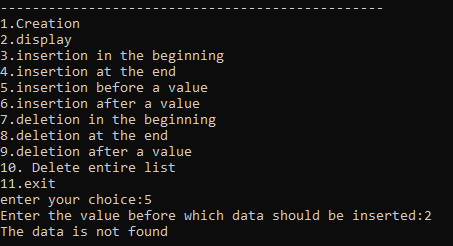
}

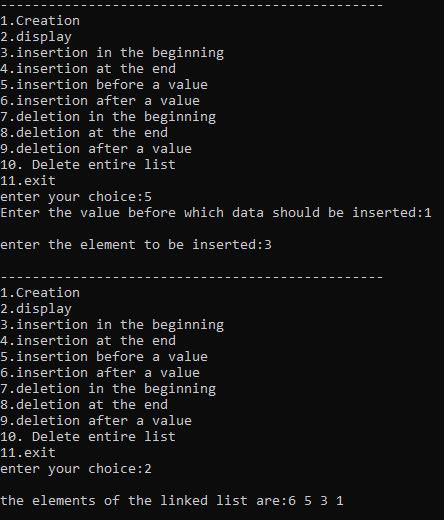
}

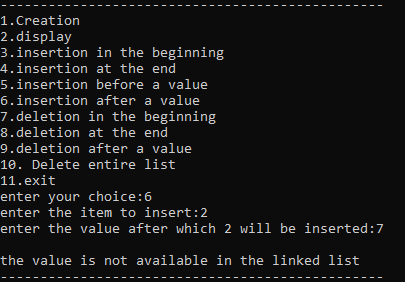
**Output :**

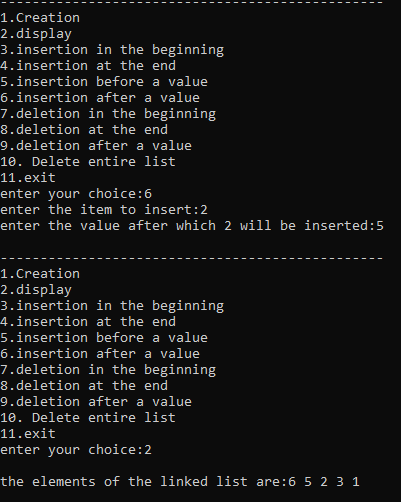
****

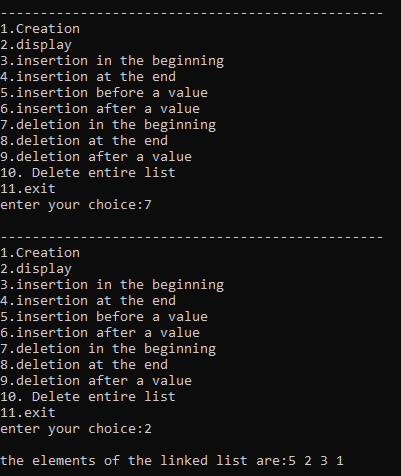
****

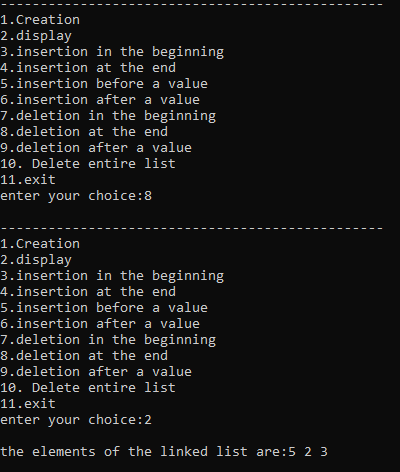
****

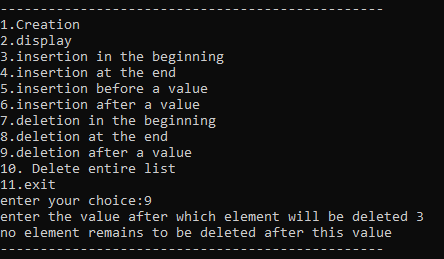
****

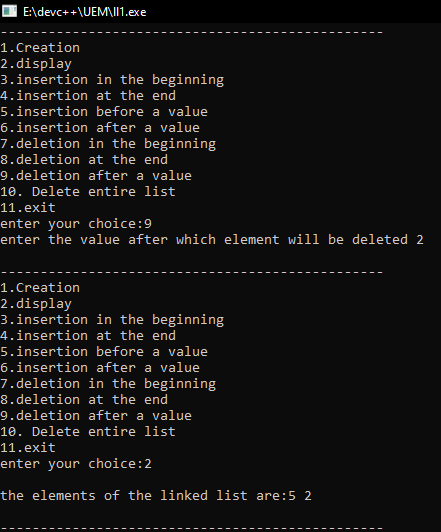
****

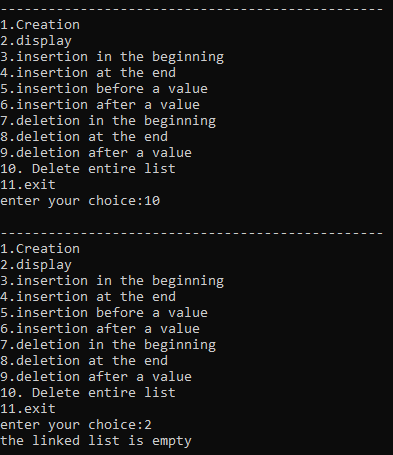
****

****

****

****

****

****

**Question 2:**  Write a Menu driven C program to accomplish the following functionalities in circular linked list.

a) Create a circular linked list.

b) Display the elements of a circular linked list.

c) Insert a node at the beginning of a circular linked list.

d) Insert a node at the end of a circular linked list.

e) Delete a node from the beginning of a circular linked list.

f) Delete a node from the end of a circular linked list.

g) Delete a node after a given node of a circular linked list.

h) Delete the entire circular linked list.

**Source Code :**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

int data;

struct node\* link;

};

struct node\* creation(struct node\* head){

struct node\* temp=(struct node\*)malloc(sizeof(struct node));

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

scanf("%d",&temp->data);

temp->link=temp;

head=temp;

return head;

}

struct node\* insert\_in\_beg(struct node\* head){

struct node\* temp=(struct node\*)malloc(sizeof(struct node));

if(temp== NULL)

printf("There is no sufficient memory");

else{

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

scanf("%d",&temp->data);

if(head==NULL)

temp->link=temp;

else{

temp->link=head;

struct node\* temp1=head;

while(temp1->link!=head){

temp1=temp1->link;

}

temp1->link=temp;

}

head=temp;

}

return head;

}

struct node\* display(struct node\* head){

if(head==NULL)

printf("the linked list is empty");

else{

struct node\* temp=head;

printf("\nthe elements are: %d ",temp->data);

temp=temp->link;

while(temp!=head){

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

temp=temp->link;

}

}

}

struct node\* insert\_at\_end(struct node\* head){

struct node\* temp=(struct node\*)malloc(sizeof(struct node));

if(temp== NULL)

printf("There is no sufficient memory");

else{

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

scanf("%d",&temp->data);

if(head==NULL){

temp->link=temp;

head=temp;

}

else{

temp->link=head;

struct node\* temp1=head;

while(temp1->link!=head){

temp1=temp1->link;

}

temp1->link=temp;

}

}

return head;

}

struct node\* del\_in\_beg(struct node\* head){

if(head==NULL)

printf("The linked list is empty");

else if(head->link==head)

head=NULL;

else{

struct node\* temp=head;

while(temp->link!=head){

temp=temp->link;

}

temp->link=head->link;

head=head->link;

}

return head;

}

struct node\* del\_at\_end(struct node\* head){

if(head==NULL)

printf("The linked list is empty");

else if(head->link==head)

head=NULL;

else{

struct node\* temp=head, \*temp1;

temp1=temp->link;

while(temp1->link!=head){

temp=temp->link;

temp1=temp->link;

}

temp->link=head;

}

return head;

}

void del\_after\_val(struct node\* head){

if(head==NULL)

printf("The linked list is empty");

else{

printf("Enter value after which node will be deleted:");

int item;

scanf("%d",&item);

struct node\* temp2=head,\*temp1;

temp1=temp2->link;

while(temp1!=head){

if(temp2->data==item){

temp2->link=temp1->link;

break;

}

temp2=temp2->link;

temp1=temp2->link;

}

if(temp1==head){

if(temp2->data==item)

printf("There is no item to be deleted after %d",item);

else

printf("The item is not found");

}

}

}

struct node\* del\_entire(struct node\* head){

head=NULL;

return head;

}

int main(){

struct node\* head=NULL;

int ch;

while(1)

{

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

printf("1.Creation \n2.display \n3.insertion in the beginning");

printf("\n4.insertion at the end\n");

printf("5.deletion in the beginning \n6.deletion at the end \n7.deletion after a value\n8. Delete entire list");

printf("\n9.exit\nenter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 3: head=insert\_in\_beg(head);

break;

case 4: head=insert\_at\_end(head);

break;

case 1: head=creation(head);

break;

case 5: head=del\_in\_beg(head);

break;

case 6: head=del\_at\_end(head);

break;

case 8: head=del\_entire(head);

break;

case 7: del\_after\_val(head);

break;

case 2: display(head);

break;

case 9: exit(0);

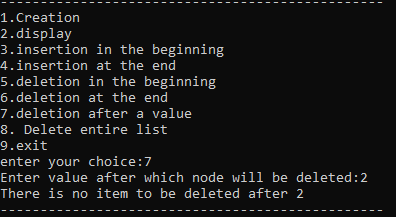
default:printf("you have entered a wrong choice");

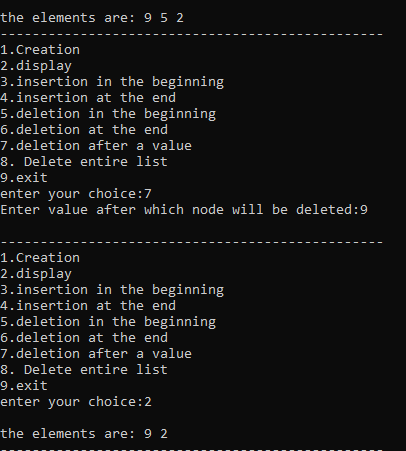
}

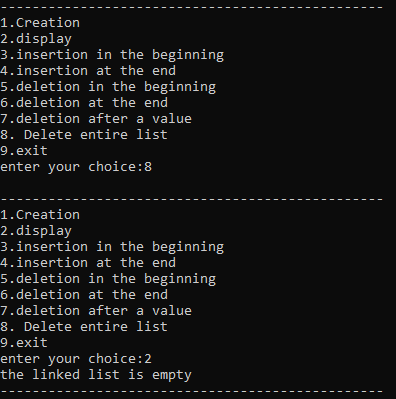
}

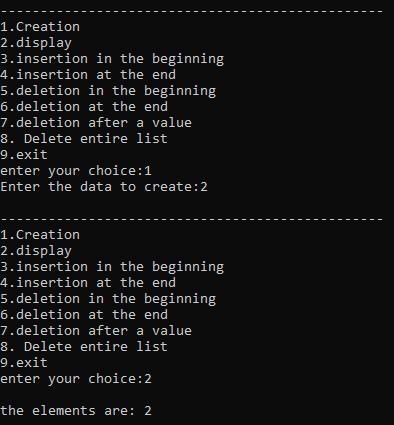
}

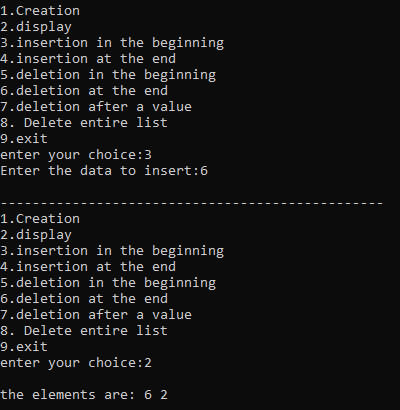
**Output :**

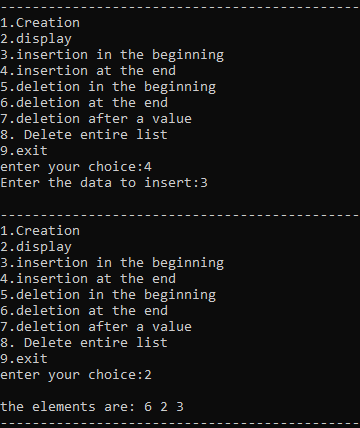
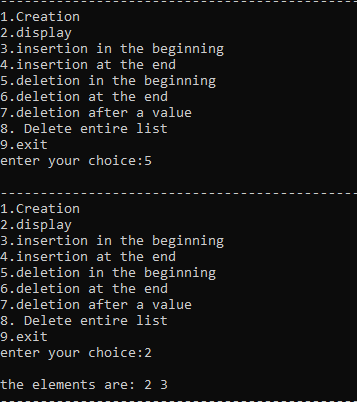
****

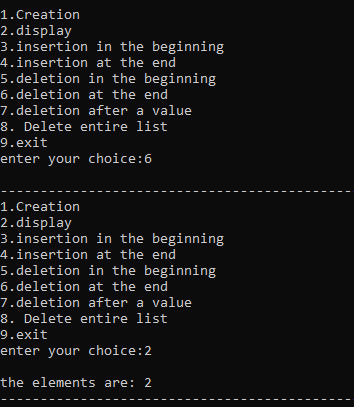
****

****

****

****

****

****

**ASSIGNMENT 4**

**Question 1:** Write a Menu driven C program to accomplish the following functionalities in doubly linked list.

a) Create a doubly linked list.

b) Display the elements of a doubly linked list.

c) Insert a node at the beginning of a doubly linked list.

d) Insert a node at the end of a doubly linked list.

e) Insert a node before a given node of a doubly linked list.

f) Insert a node after a given node of a doubly linked list.

g) Delete a node from the beginning of a doubly linked list.

h) Delete a node from the end of a doubly linked list.

i) Delete a node after a given node of a doubly linked list.

j) Delete the entire doubly linked list.

**Source Code :**

#include <stdio.h>

#include <stdlib.h>

struct dll

{

int data;

struct dll \*next;

struct dll \*prev;

};

struct dll \*head =NULL;

void ins\_beg();

void ins\_end();

void ins\_after();

void ins\_before();

void display();

void del\_beg();

void del\_end();

void del\_after();

int main()

{

int n,k;

char c;

do

{

printf("\n1. insert at begining\n2. insert at end\n3. insert after a node \n4. insert before a node\n5. display\n6. delete from begining");

printf("\n7. delete from end \n8. delete after a node");

printf("\n\n Enter your choice ");

scanf("%d",&n);

switch(n)

{

case 1: ins\_beg();

break;

case 2: ins\_end();

break;

case 3: printf("\n enter the key after which the node will be inserted ");

scanf("%d",&k);

ins\_after(k);

break;

case 4: printf("\n enter the key before which the node will be inserted ");

scanf("%d",&k);

ins\_before(k);

break;

case 5: display();

break;

case 6: del\_beg();

break;

case 7: del\_end();

break;

case 8: del\_after();

break;

default : printf("\n please enter a valid choice ");

}

printf("\n do you want to cont. (Y/y)");

fflush(stdin);

scanf("%c",&c);

}while(c=='Y'||c=='y');

}

void ins\_beg()

{

struct dll \*newnode;

newnode = (struct dll\*)malloc(sizeof(struct dll));

printf("\n enter the data to be inserted :");

scanf("%d",&newnode->data);

newnode->prev=NULL;

if (head ==NULL)

{

newnode->next=NULL;

head = newnode;

}

else

{

newnode->next=head;

head->prev=newnode;

head=newnode;

}

}

void ins\_end()

{

struct dll \*newnode,\*ptr;

newnode=(struct dll \*)malloc(sizeof(struct dll));

printf("\n enter the data to be inserted : ");

scanf("%d", &newnode->data);

if(head == NULL)

{

newnode->next=NULL;

head = newnode;

newnode->prev=NULL;

}

else

{

ptr=head;

while (ptr->next !=NULL)

ptr = ptr->next;

ptr->next = newnode;

newnode->prev = ptr;

newnode->next=NULL;

}

}

void ins\_after(int k)

{

struct dll \*newnode,\*ptr;

newnode=(struct dll \*)malloc(sizeof(struct dll));

printf("\n enter the data to be inserted : ");

scanf("%d", &newnode->data);

ptr=head;

while (ptr->next !=NULL && ptr->data !=k)

ptr=ptr->next;

if(ptr->data == k && ptr->next==NULL)

{

ptr->next=newnode;

newnode->next=NULL;

newnode->prev=ptr;

}

else if(ptr->data==k && ptr->next!=NULL)

{

newnode->next=ptr->next;

newnode->prev=ptr;

ptr->next->prev=newnode;

ptr->next=newnode;

}

else

printf("\n no such data found");

}

void ins\_before(int k)

{

struct dll \*newnode,\*ptr;

newnode=(struct dll \*)malloc(sizeof(struct dll));

printf("\n enter the data to be inserted : ");

scanf("%d", &newnode->data);

ptr=head;

while (ptr->next !=NULL && ptr->data !=k)

ptr=ptr->next;

if(ptr->data == k && ptr->next==NULL)

{

ptr->next=newnode;

newnode->next=NULL;

newnode->prev=ptr;

}

else if(ptr->data==k && ptr->prev!=NULL)

{

newnode->next=ptr;

newnode->prev=ptr->prev;

ptr->prev->next=newnode;

ptr->prev=newnode;

}

else

printf("\n no such data found");

}

void display()

{

struct dll \*ptr = head;

if(ptr==NULL)

printf("\n EMPTY LIST");

while(ptr != NULL)

{

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

ptr = ptr->next;

}

}

void del\_beg()

{

struct dll \*ptr;

if (head ==NULL)

printf("\n empty list");

else

{

if(head->next ==NULL)

{

printf("\n data deleted = %d",head->data);

head=NULL;

free(head);

}

else

{

ptr =head;

head=head->next;

head->prev=NULL;

printf("\n deta deleted = %d",ptr->data);

free(ptr);

}

}

}

void del\_end()

{

struct dll \*ptr,\*pptr;

if (head ==NULL)

printf("\n empty list");

else

{

if(head->next ==NULL)

{

printf("\n data deleted = %d",head->data);

head=NULL;

free(head);

}

else

{

ptr=head;

while(ptr->next !=NULL)

{

pptr=ptr;

ptr=ptr->next;

}

printf("\n deleted data = %d",ptr->data);

pptr->next=NULL;

free(ptr);

}

}

}

void del\_after()

{

struct dll \*ptr,\*temp;

int k;

printf("\n after delete");

scanf("%d",&k);

ptr=head;

while(ptr->data != k )

ptr=ptr->next;

if(ptr->next==NULL)

{

printf("\n empty list");

}

else if(ptr->next->next==NULL)

{

ptr->next=NULL;

}

else

{

temp=ptr->next;

ptr->next=temp->next;

temp->next->prev=ptr;

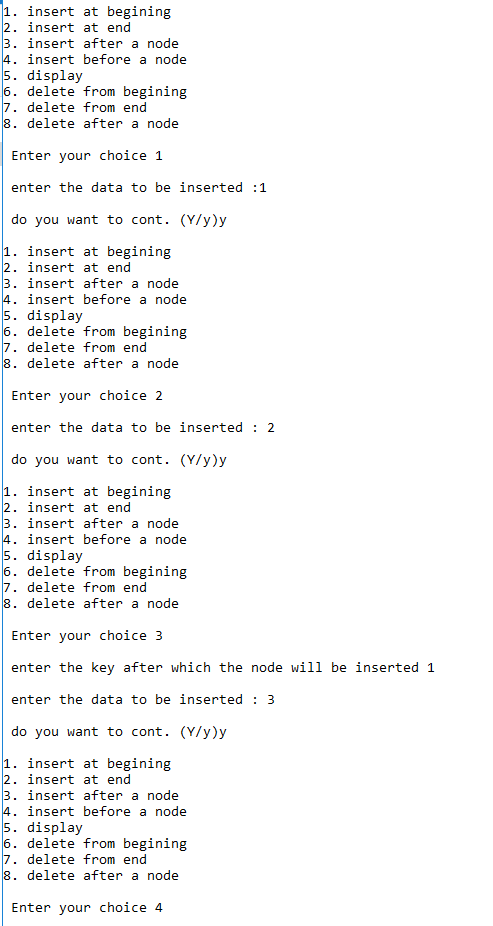
free(temp);

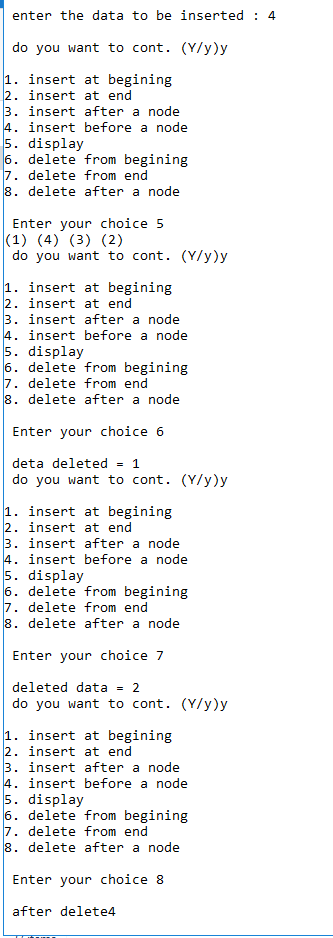
printf("\n node deleted");

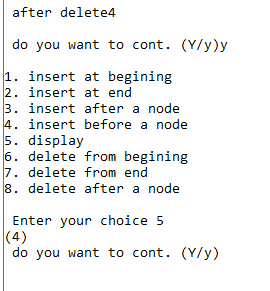
}

}

**Output :**

****





**Question- 2:** Write a Menu driven C program to accomplish the following functionalities in circular doubly linked list.

a) Create a circular doubly linked list.

b) Display the elements of a circular doubly linked list.

c) Insert a node at the beginning of a circular doubly linked list.

d) Insert a node at the end of a circular doubly linked list.

e) Delete a node from the beginning of a circular doubly linked list.

f) Delete a node from the end of a circular doubly linked list.

g) Delete a node after a given node of a circular doubly linked list.

h) Delete the entire circular doubly linked list.

**Source Code :**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

struct Node\* prev;

};

struct Node\* createNode(int data);

void displayList(struct Node\* head);

struct Node\* insertAtBeginning(struct Node\* head, int data);

struct Node\* insertAtEnd(struct Node\* head, int data);

struct Node\* deleteFromBeginning(struct Node\* head);

struct Node\* deleteFromEnd(struct Node\* head);

struct Node\* deleteAfterNode(struct Node\* head, struct Node\* target);

struct Node\* deleteEntireList(struct Node\* head);

int main() {

struct Node\* head = NULL;

int choice, data;

struct Node\* targetNode;

do {

// Display menu

printf("\n1. Create a circular doubly linked list");

printf("\n2. Display elements");

printf("\n3. Insert at the beginning");

printf("\n4. Insert at the end");

printf("\n5. Delete from the beginning");

printf("\n6. Delete from the end");

printf("\n7. Delete after a given node");

printf("\n8. Delete the entire list");

printf("\n9. Exit");

printf("\nEnter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter data for the first node: ");

scanf("%d", &data);

head = createNode(data);

break;

case 2:

displayList(head);

break;

case 3:

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

scanf("%d", &data);

head = insertAtBeginning(head, data);

break;

case 4:

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

scanf("%d", &data);

head = insertAtEnd(head, data);

break;

case 5:

head = deleteFromBeginning(head);

break;

case 6:

head = deleteFromEnd(head);

break;

case 7:

printf("Enter data of the node after which you want to delete: ");

scanf("%d", &data);

targetNode = head;

while (targetNode != NULL && targetNode->data != data) {

targetNode = targetNode->next;

}

if (targetNode != NULL) {

head = deleteAfterNode(head, targetNode);

} else {

printf("Node with given data not found.\n");

}

break;

case 8:

head = deleteEntireList(head);

break;

case 9:

printf("Exiting program.\n");

break;

default:

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

}

} while (choice != 9);

return 0;

}

struct Node\* createNode(int data) {

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

newNode->data = data;

newNode->next = newNode;

newNode->prev = newNode;

return newNode;

}

void displayList(struct Node\* head) {

if (head == NULL) {

printf("List is empty.\n");

return;

}

struct Node\* current = head;

do {

printf("%d <-> ", current->data);

current = current->next;

} while (current != head);

printf("(head)\n");

}

struct Node\* insertAtBeginning(struct Node\* head, int data) {

struct Node\* newNode = createNode(data);

if (head == NULL) {

return newNode;

}

newNode->next = head;

newNode->prev = head->prev;

head->prev->next = newNode;

head->prev = newNode;

return newNode;

}

struct Node\* insertAtEnd(struct Node\* head, int data) {

struct Node\* newNode = createNode(data);

if (head == NULL) {

return newNode;

}

newNode->next = head;

newNode->prev = head->prev;

head->prev->next = newNode;

head->prev = newNode;

return head;

}

struct Node\* deleteFromBeginning(struct Node\* head) {

if (head == NULL) {

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

return NULL;

}

if (head->next == head) {

free(head);

return NULL;

}

struct Node\* newHead = head->next;

newHead->prev = head->prev;

head->prev->next = newHead;

free(head);

return newHead;

}

struct Node\* deleteFromEnd(struct Node\* head) {

if (head == NULL) {

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

return NULL;

}

if (head->next == head) {

free(head);

return NULL;

}

struct Node\* newEnd = head->prev;

newEnd->next = head;

head->prev = newEnd->prev;

free(newEnd);

return head;

}

struct Node\* deleteAfterNode(struct Node\* head, struct Node\* target) {

if (head == NULL || target == NULL) {

printf("Invalid operation.\n");

return head;

}

if (target->next == target) {

printf("Cannot delete the only remaining node.\n");

return head;

}

struct Node\* nodeToDelete = target->next;

target->next = nodeToDelete->next;

nodeToDelete->next->prev = target;

free(nodeToDelete);

return head;

}

struct Node\* deleteEntireList(struct Node\* head) {

struct Node\* current = head;

struct Node\* nextNode;

while (current != NULL) {

nextNode = current->next;

free(current);

current = nextNode;

if (current == head) {

break;

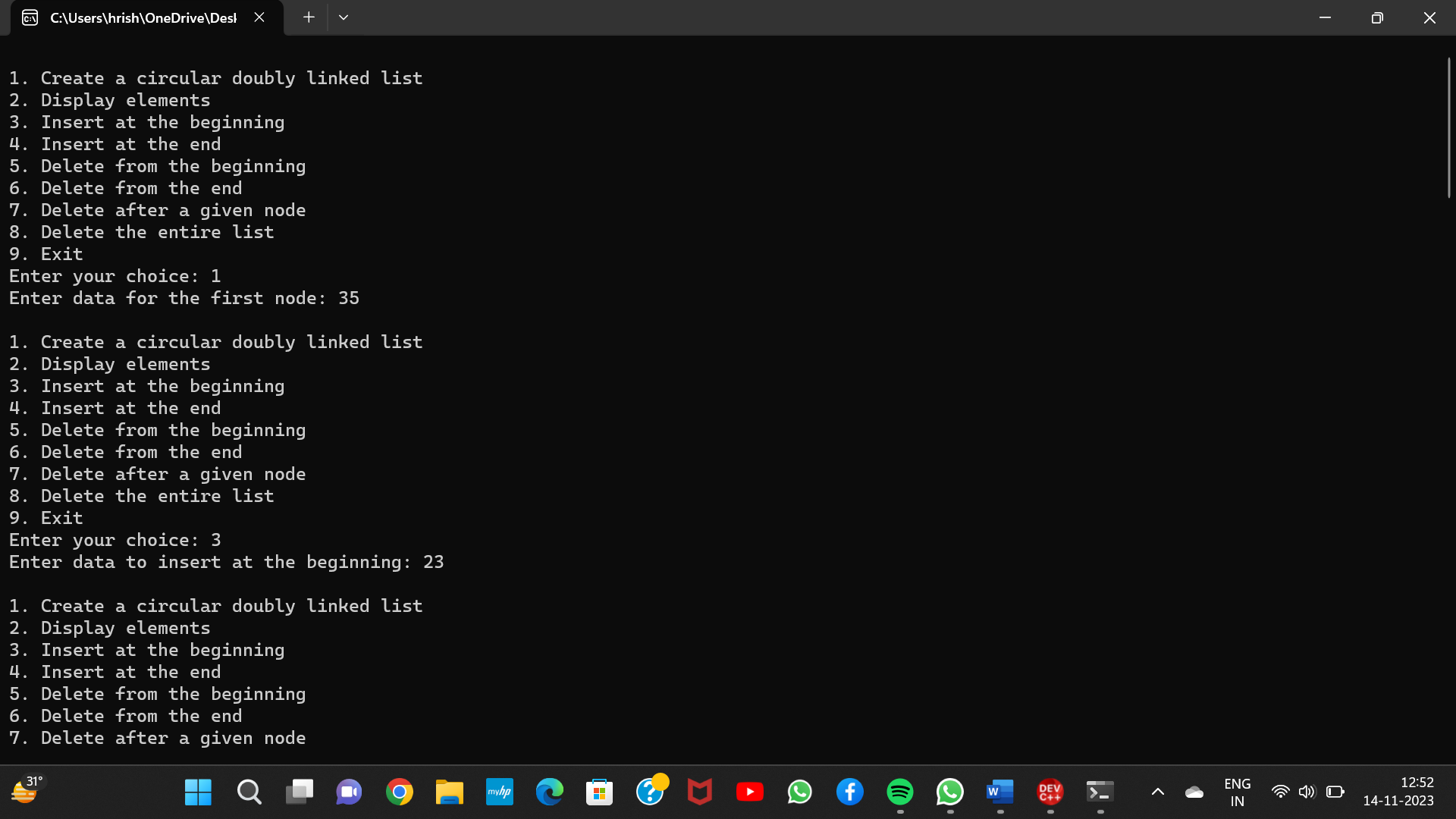
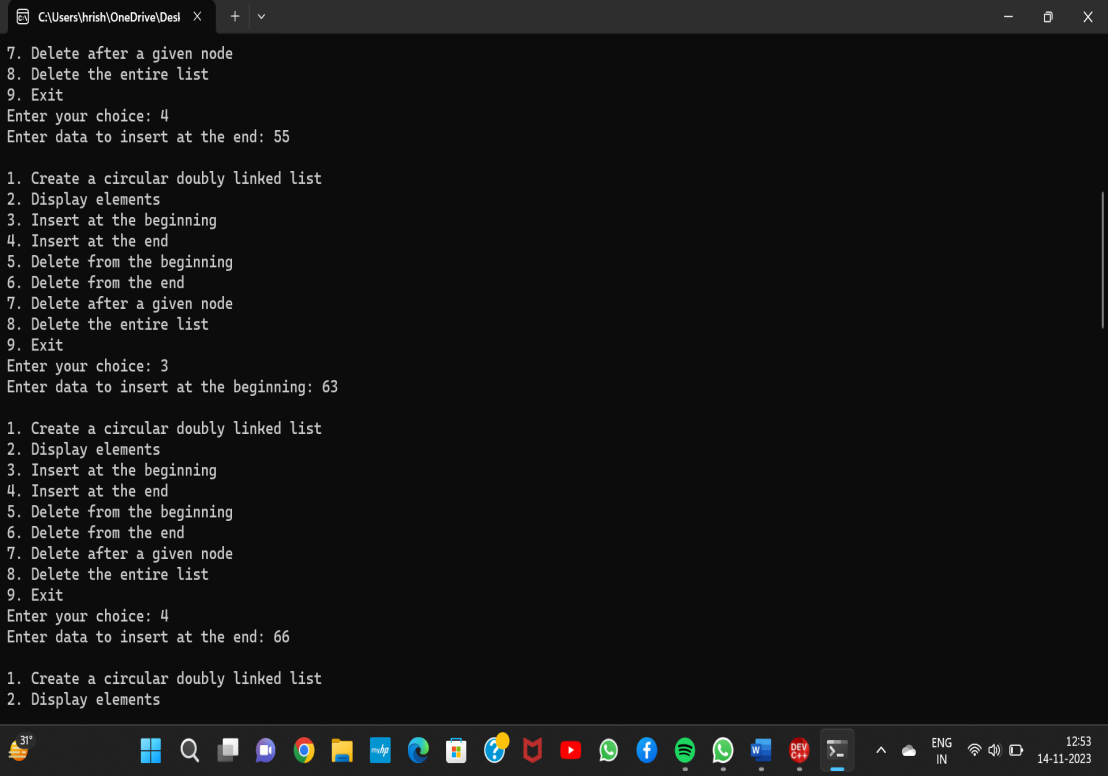
}

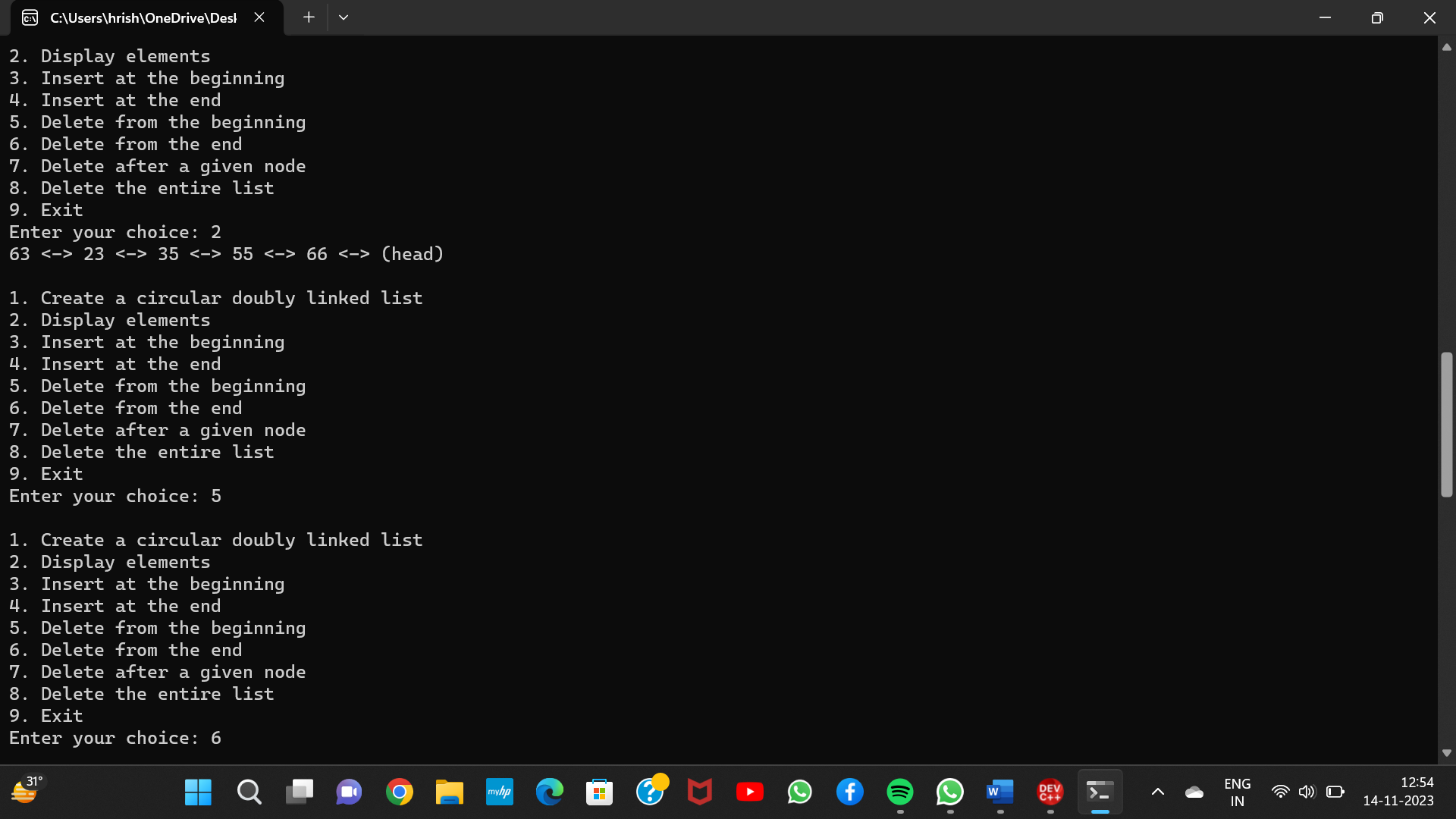
}

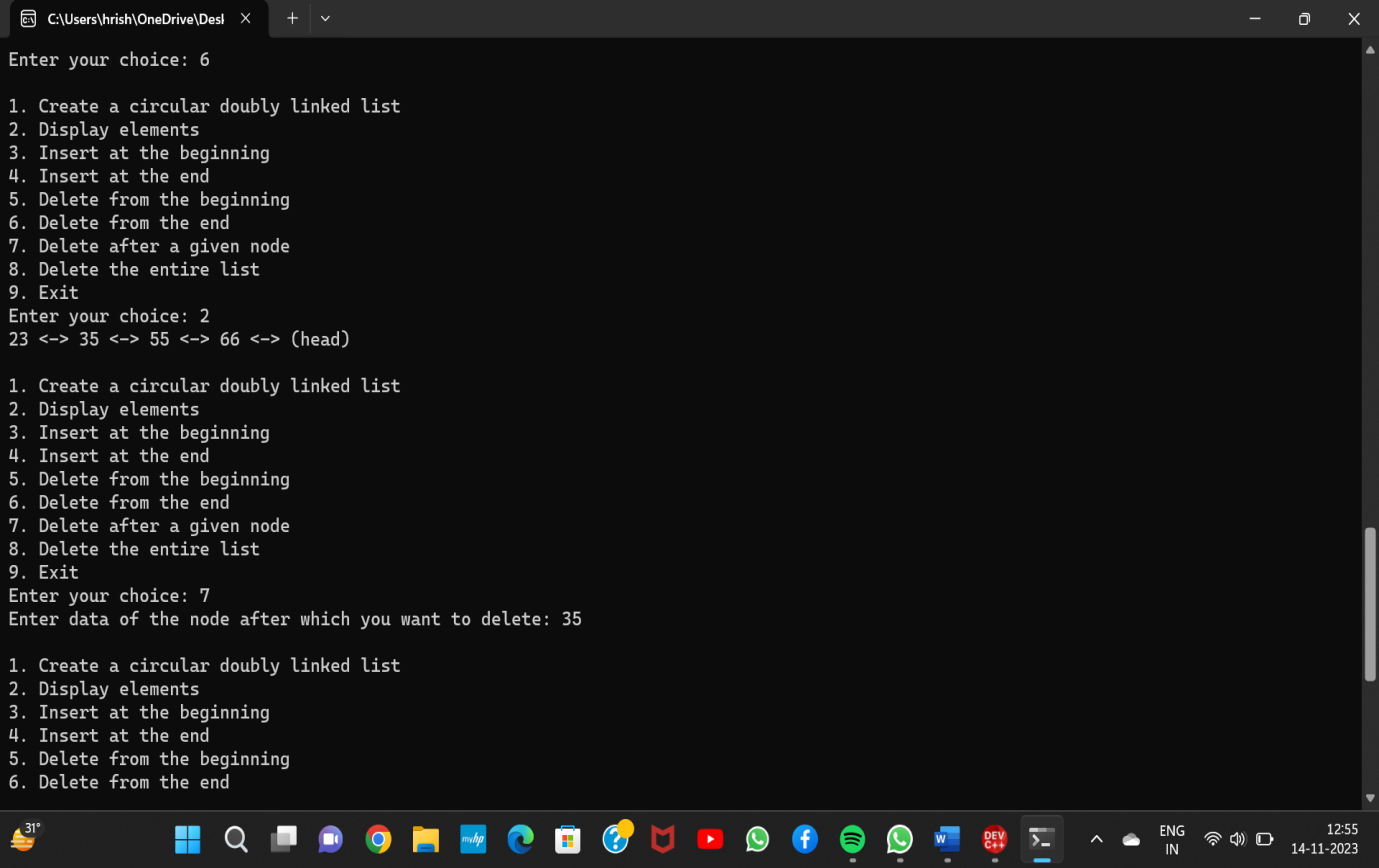
return NULL;

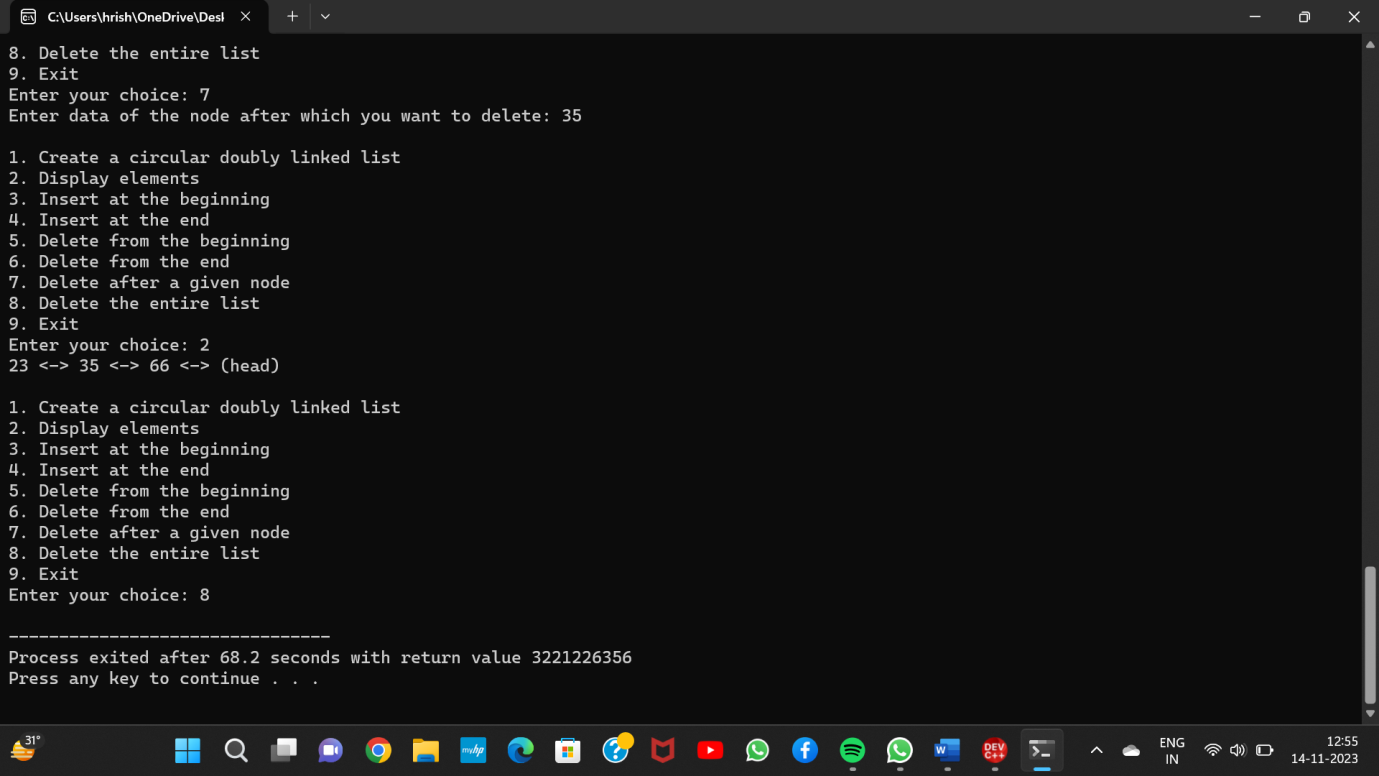
}

**Output:**









**ASSIGNMENT 5**

**Question 1:** Write a Menu driven C program to accomplish the following functionalities in Queue using an Array:

a. Insert an element into the queue using an array (Enqueue Operation).

b. Delete an element from the queue using an array (Dequeue Operation).

c. Return the value of the FRONT element of the queue (without deleting it from the

queue) using an array (Peep operation).

d. Display the elements of a queue using an array.

**Source Code :**

#include <stdio.h>

#include <stdlib.h>

#define MAX 10

int queue[MAX];

int front = -1, rear = -1;

void enqueue(int element) {

if (rear == MAX - 1) {

printf("Queue is full. Cannot enqueue.\n");

} else {

if (front == -1) {

front = 0;

}

rear++;

queue[rear] = element;

printf("%d has been enqueued.\n", element);

}

}

void dequeue() {

if (front == -1) {

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

} else {

printf("%d has been dequeued.\n", queue[front]);

if (front == rear) {

front = rear = -1;

} else {

front++;

}

}

}

int peek() {

if (front == -1) {

printf("Queue is empty. No front element to peek.\n");

return -1; // Return an error value

} else {

return queue[front];

}

}

void display() {

if (front == -1) {

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

} else {

printf("Queue elements: ");

for (int i = front; i <= rear; i++) {

printf("%d ", queue[i]);

}

printf("\n");

}

}

int main() {

int choice, element;

while (1) {

printf("\nMenu:\n");

printf("1. Enqueue\n");

printf("2. Dequeue\n");

printf("3. Peek\n");

printf("4. Display\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter element to enqueue: ");

scanf("%d", &element);

enqueue(element);

break;

case 2:

dequeue();

break;

case 3:

element = peek();

if (element != -1) {

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

}

break;

case 4:

display();

break;

case 5:

exit(0);

default:

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

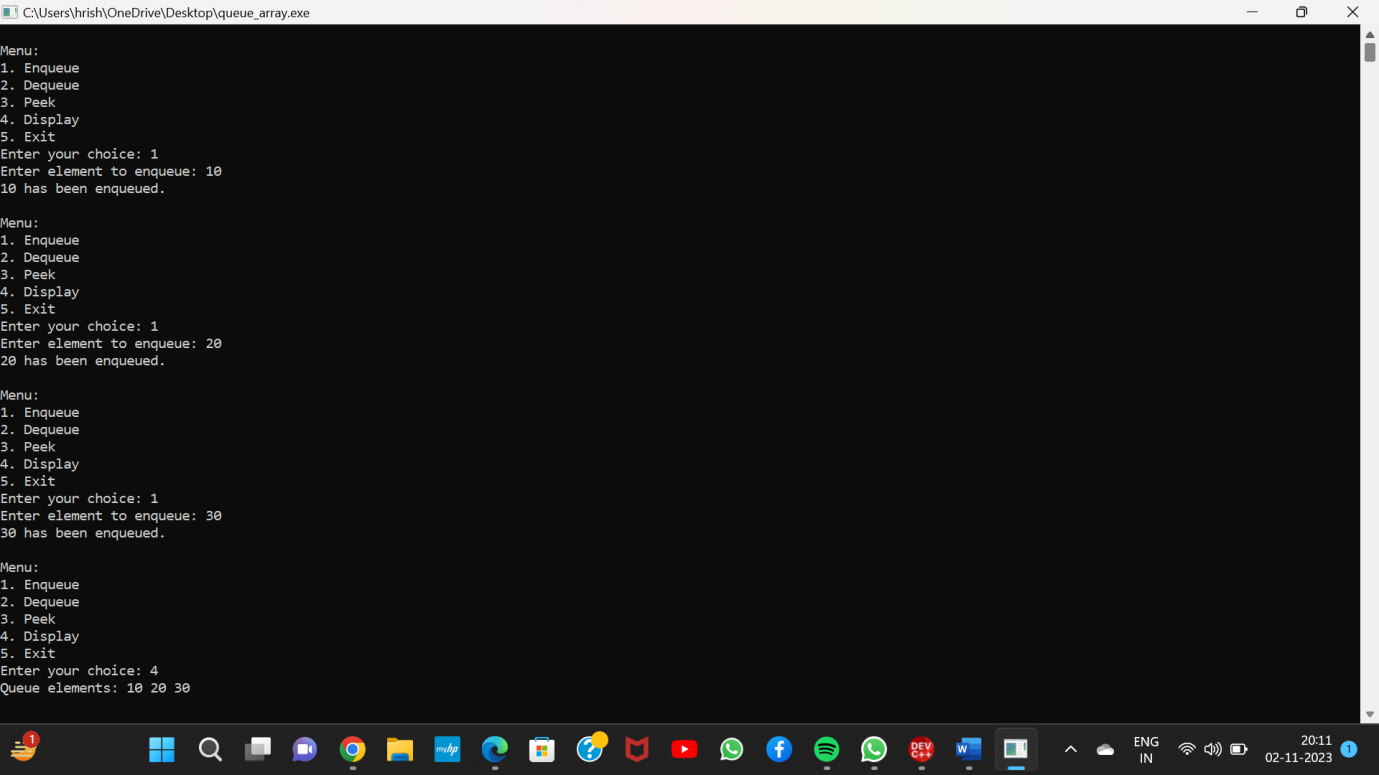
}

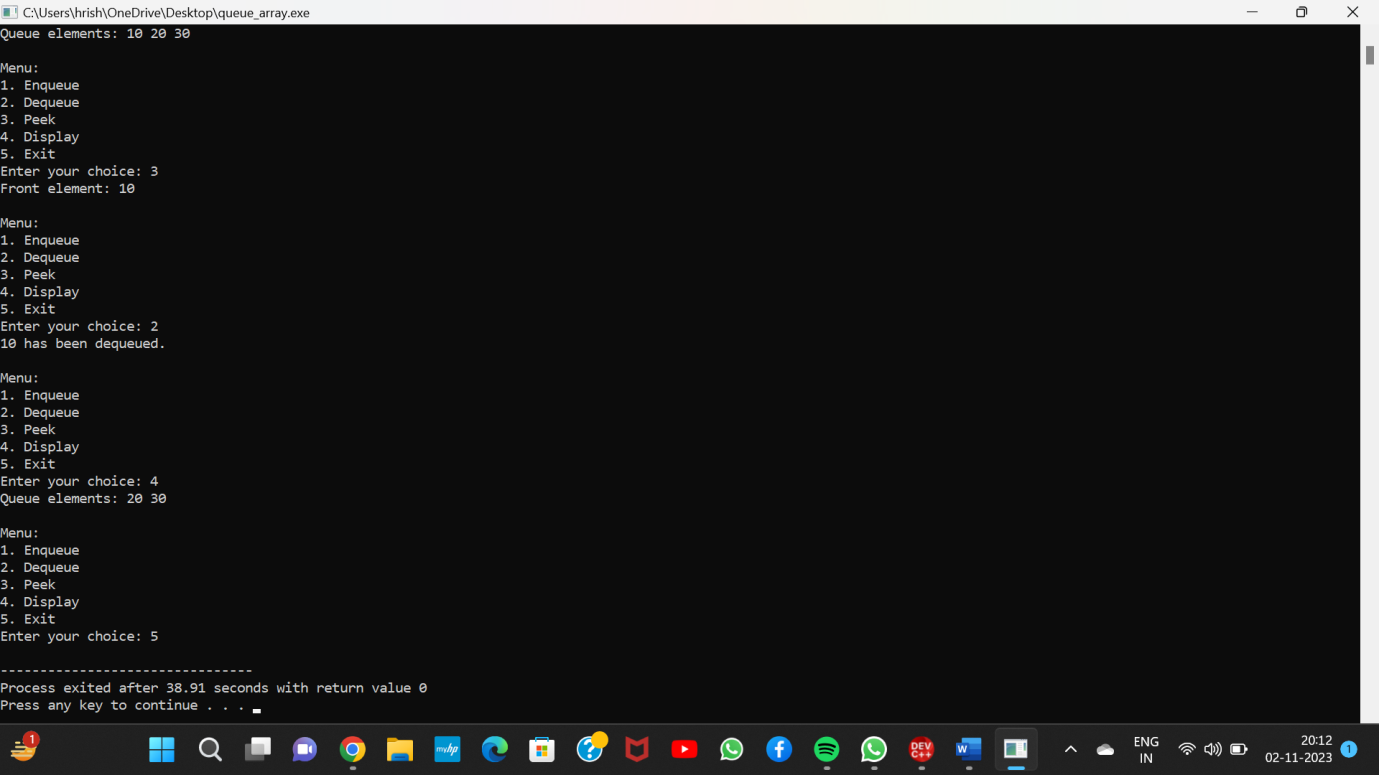
}

return 0;

}

**Output :**





**Question 2:** Write a Menu driven C program to accomplish the following functionalities in Queue using Linked List:

e. Insert an element into the queue using a Linked List (Enqueue Operation).

f. Delete an element from the queue using a Linked List (Dequeue Operation).

g. Return the value of the FRONT element of the queue (without deleting it from the

queue) using a Linked List (Peep operation).

h. Display the elements of a queue using a Linked List.

**Source Code :**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

struct Node\* front = NULL;

struct Node\* rear = NULL;

void enqueue(int element) {

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

newNode->data = element;

newNode->next = NULL;

if (rear == NULL) {

front = rear = newNode;

} else {

rear->next = newNode;

rear = newNode;

}

printf("%d has been enqueued.\n", element);

}

void dequeue() {

if (front == NULL) {

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

} else {

struct Node\* temp = front;

front = front->next;

free(temp);

if (front == NULL) {

rear = NULL;

}

}

}

int peek() {

if (front == NULL) {

printf("Queue is empty. No front element to peek.\n");

return -1; // Return an error value

} else {

return front->data;

}

}

void display() {

if (front == NULL) {

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

} else {

struct Node\* current = front;

printf("Queue elements: ");

while (current != NULL) {

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

current = current->next;

}

printf("\n");

}

}

int main() {

int choice, element;

while (1) {

printf("\nMenu:\n");

printf("1. Enqueue\n");

printf("2. Dequeue\n");

printf("3. Peek\n");

printf("4. Display\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter element to enqueue: ");

scanf("%d", &element);

enqueue(element);

break;

case 2:

dequeue();

break;

case 3:

element = peek();

if (element != -1) {

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

}

break;

case 4:

display();

break;

case 5:

exit(0);

default:

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

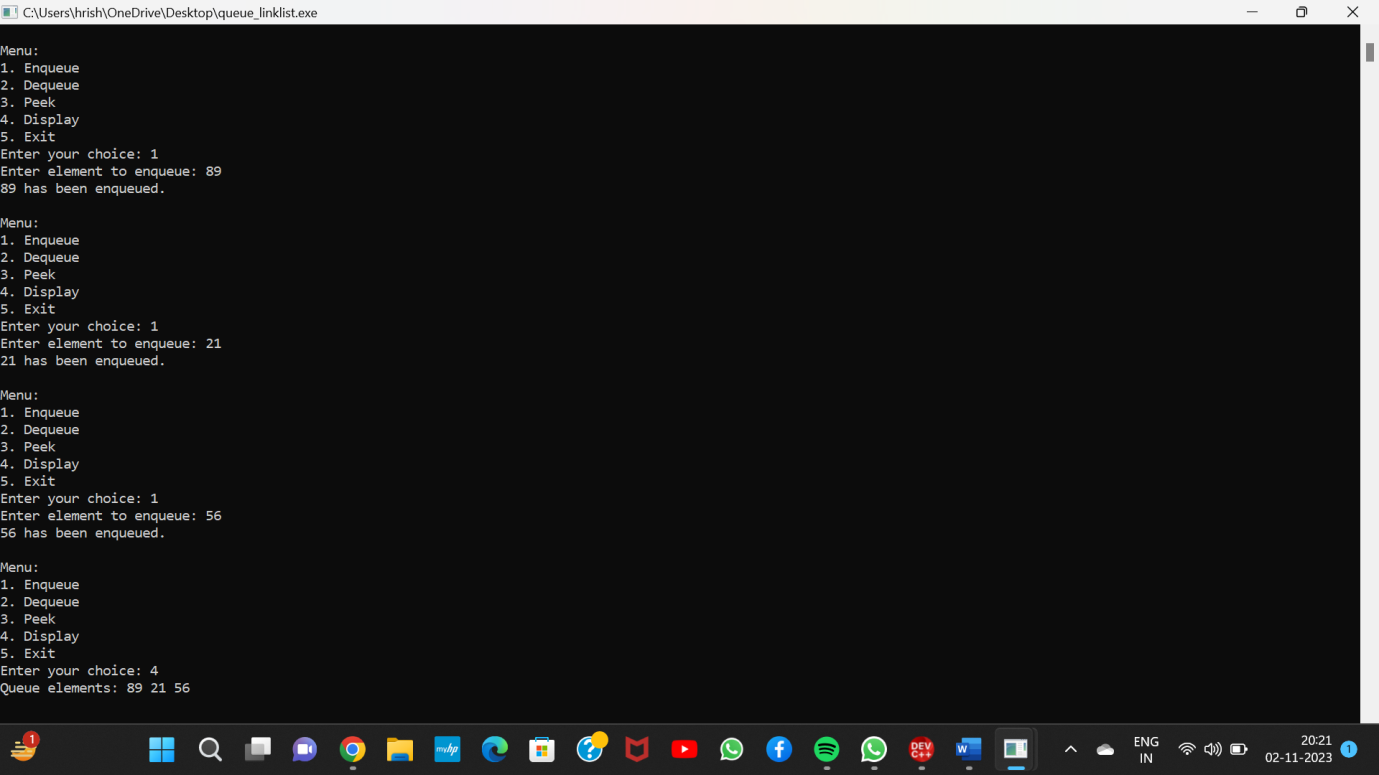
}

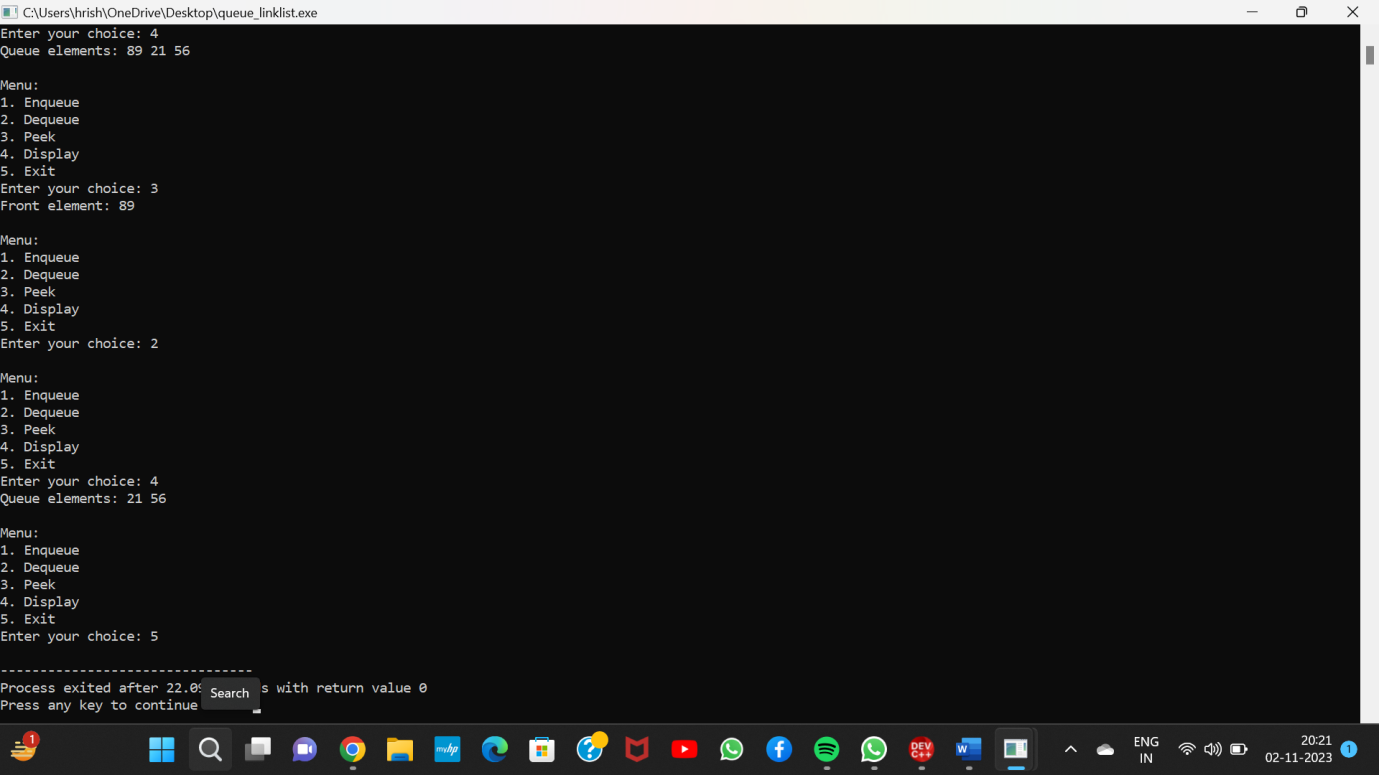
}

return 0;

}

**Output :**





**Question 3:**  Write a Menu driven C program to accomplish the following functionalities in Circular Queue using Array:

i. Insert an element into the circular queue.

j. Delete an element from the circular queue.

k. Return the value of the FRONT element of the circular queue (without deleting it

from the queue).

l. Display the elements of a circular queue using the circular queue.

**Source Code :**

#include <stdio.h>

#define MAX\_SIZE 10

int circularQueue[MAX\_SIZE];

int front = -1, rear = -1;

int isFull() {

return (front == 0 && rear == MAX\_SIZE - 1) || (front == rear + 1);

}

int isEmpty() {

return front == -1;

}

void enqueue(int element) {

if (isFull()) {

printf("Queue is full. Cannot enqueue.\n");

} else {

if (front == -1) {

front = 0;

}

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

circularQueue[rear] = element;

printf("%d has been enqueued.\n", element);

}

}

void dequeue() {

if (isEmpty()) {

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

} else {

int removedElement = circularQueue[front];

if (front == rear) {

front = rear = -1;

} else {

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

}

printf("%d has been dequeued.\n", removedElement);

}

}

int peek() {

if (isEmpty()) {

printf("Queue is empty. No front element to peek.\n");

return -1; // Return an error value

} else {

return circularQueue[front];

}

}

void display() {

if (isEmpty()) {

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

} else {

int i = front;

printf("Queue elements: ");

do {

printf("%d ", circularQueue[i]);

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

} while (i != (rear + 1) % MAX\_SIZE);

printf("\n");

}

}

int main() {

int choice, element;

while (1) {

printf("\nMenu:\n");

printf("1. Enqueue\n");

printf("2. Dequeue\n");

printf("3. Peek\n");

printf("4. Display\n");

printf("5. Exit\n");

printf("Enter your choice: ");

int option;

scanf(" %d", &option);

switch (option) {

case 1:

printf("Enter element to enqueue: ");

scanf("%d", &element);

enqueue(element);

break;

case 2:

dequeue();

break;

case 3:

element = peek();

if (element != -1) {

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

}

break;

case 4:

display();

break;

case 5:

return 0;

default:

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

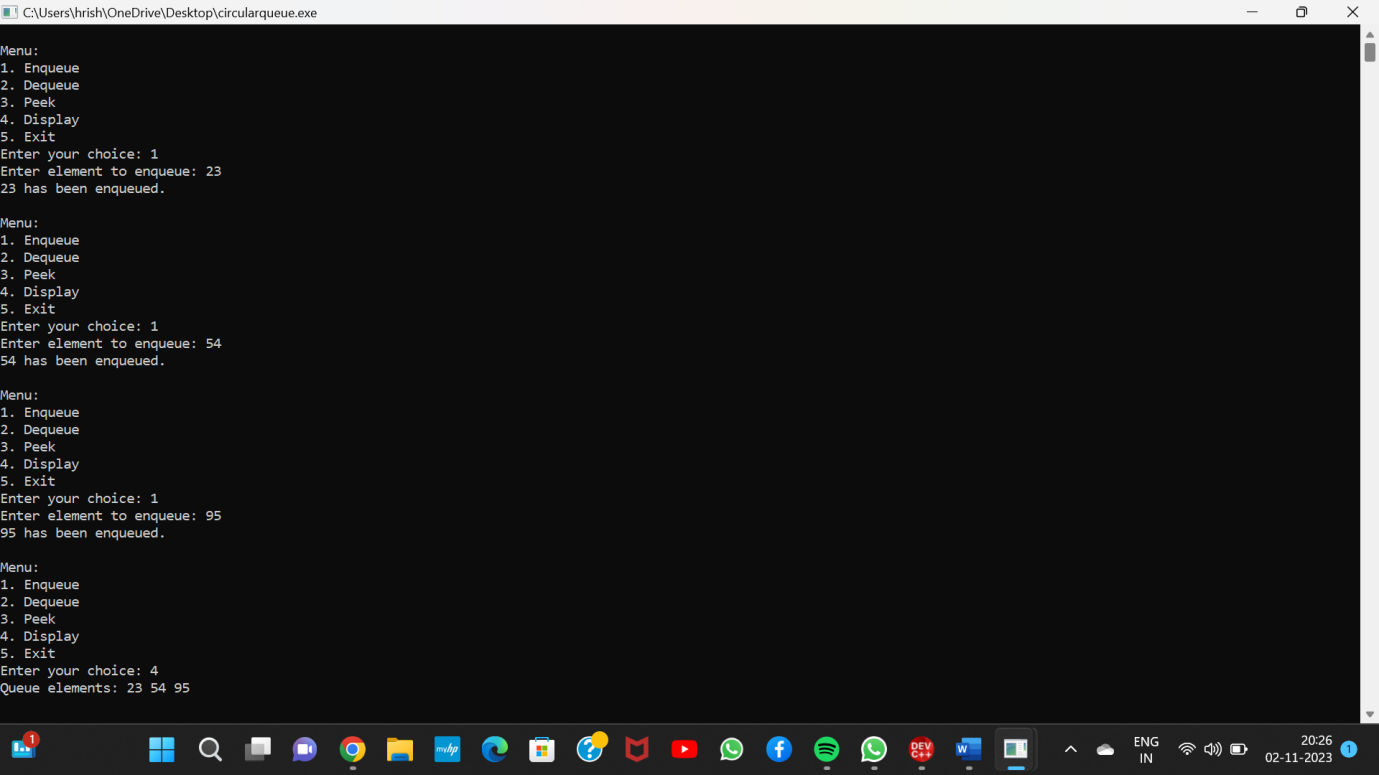
}

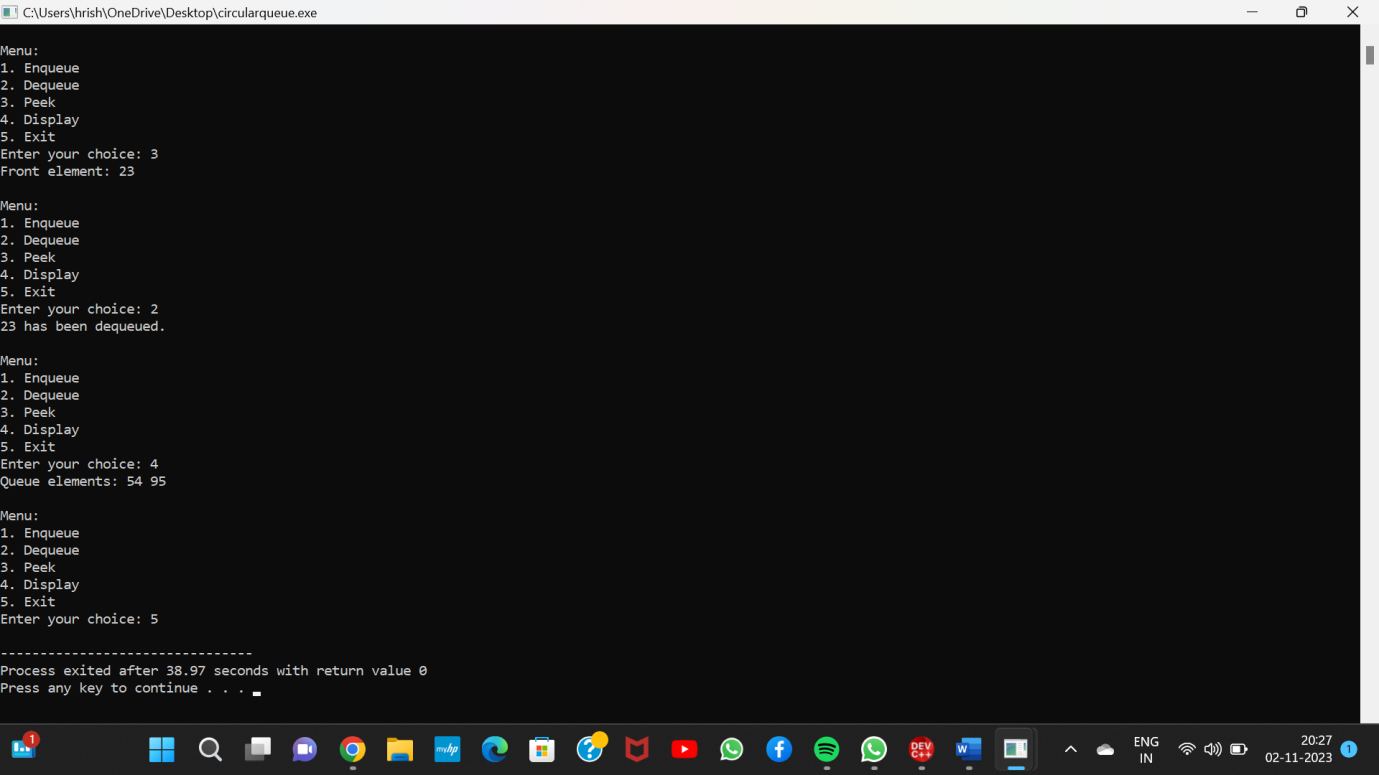
}

return 0;

}

**Output :**





**ASSIGNMENT 6**

**Question 1:** Write a Menu driven C program to accomplish the following functionalities in Stack using an Array:

a. Insert an element into the stack using an array (Push Operation).

b. Delete an element from the stack using an array (Pop Operation).

c. Return the value of the TOP element of the stack(without deleting it from the

stack) using an array.

d. Display the elements of a stack using an array.

**Source Code :**

#include<stdio.h>

int stack[100],choice,n,top,x,i;

void push(void);

void pop(void);

void display(void);

int main()

{

//clrscr();

top=-1;

printf("\n Enter the size of STACK[MAX=100]:");

scanf("%d",&n);

printf("\n\t STACK OPERATIONS USING ARRAY");

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

printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");

do

{

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

scanf("%d",&choice);

switch(choice)

{

case 1:

{

push();

break;

}

case 2:

{

pop();

break;

}

case 3:

{

display();

break;

}

case 4:

{

printf("\n\t EXIT POINT ");

break;

}

default:

{

printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");

}

}

}

while(choice!=4);

return 0;

}

void push()

{

if(top>=n-1)

{

printf("stack is overflow:");

}

else

{

printf("enter a item:");

scanf("%d",&x);

top=top+1;

stack[top]=x;

}

}

void pop()

{

if(top==-1)

{

printf("stack is underflow:");

}

else

{

printf("%d is the deleted item",stack[top]);

top=top-1**;**

}

}

void display()

{

if(top==-1)

{

printf("no items to display:");

}

else

{

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

{

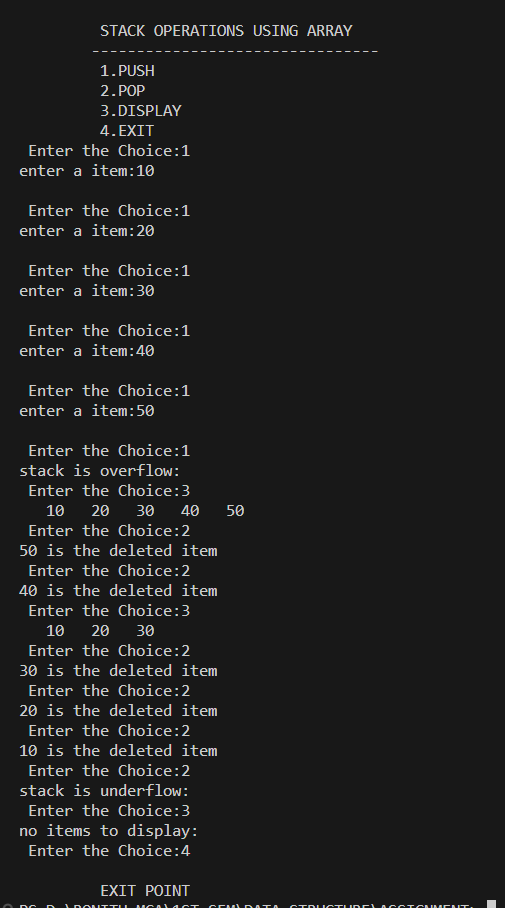
printf("%5d",stack[i]);

}

}

}

**Output :**

****

**Question 2:** Write a Menu driven C program to accomplish the following functionalities in Stack using an Linked List:

a. Insert an element into the stack using an Linked List(Push Operation).

b. Delete an element from the stack using an Linked List(Pop Operation).

c. Return the value of the TOP element of the Stack using Linked List(without deleting it from the

stack) using an array.

d. Display the elements of a stack using an Linked List.

**Source Code :**

#include<stdio.h>

struct node{

int data;

struct node \*next;

};

struct node \*top=NULL;

int choice,n,val;

void push(){

printf("enter a item:");

scanf("%d",&val);

struct node \*newnode=(struct node \*)malloc(sizeof(struct node));

newnode->data=val;

if(top==NULL){

top=newnode;

newnode->next=NULL;

}

else{

newnode->next=top;

top=newnode;

}

}

void peek(){

printf("\n%d is current top element",top->data);

}

void pop(){

if(top==NULL){

printf("\nstack is empty...");

}

else{

printf("\npopped item is->%d",top->data);

top=top->next;

}

}

void display(){

struct node \*temp=top;

if(top==NULL){

printf("\nstack is empty...");

}

else{

printf("\n List is->");

while(temp!=NULL){

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

temp=temp->next;

}

}

}

int main(){

printf("\n Enter the size of STACK[MAX=100]:");

scanf("%d",&n);

printf("\n\t STACK OPERATIONS USING ARRAY");

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

printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");

do

{

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

scanf("%d",&choice);

switch(choice)

{

case 1:

{

push();

break;

}

case 2:

{

pop();

break;

}

case 3:

{

display();

break;

}

case 4:

{

printf("\n\t EXIT POINT ");

break;

}

default:

{

printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");

}

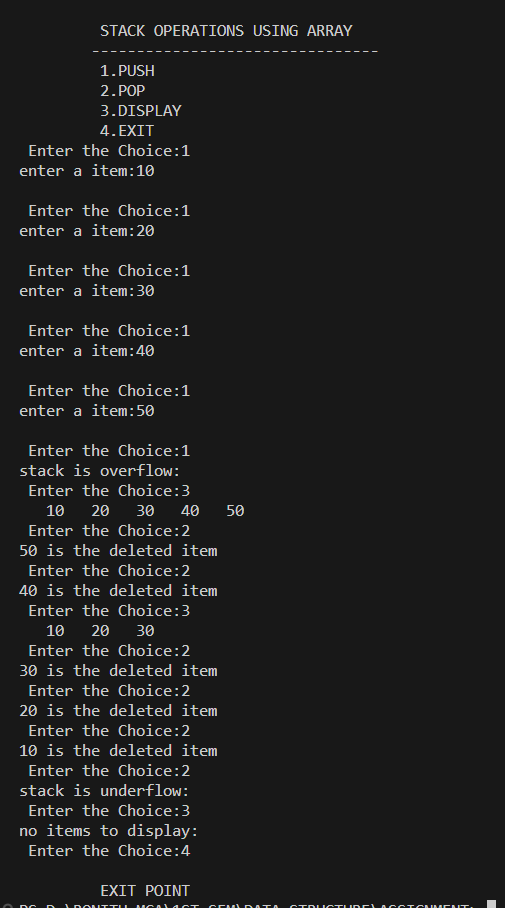
}

}

while(choice!=4);

}

**Output :**

****

**Question 3:** Write a C program to convert an infix expression to its equivalent postfix expression.

**Source Code :**

#include <stdio.h>

#include <string.h>

char stack[100];

int top = -1;

void Push(char item)

{

top += 1;

stack[top] = item;

}

char Peek()

{

return stack[top];

}

char Pop()

{

char item = stack[top];

top -= 1;

return item;

}

int main()

{

char infix[100], postfix[100];

printf("Enter the infix expression->");

gets(infix);

int i = 0, idx = 0;

while (infix[i] != '\0')

{

if ((infix[i] >= 'A' && infix[i] <= 'Z') || (infix[i] >= 'a' && infix[i] <= 'z'))

{

postfix[idx] = infix[i];

idx++;

// printf("%c----\n",infix[i]);

}

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

postfix[idx] = infix[i];

idx++;

}

else

{

if (top == -1)

{

Push(infix[i]);

}

else{

if ((infix[i] == '\*' || infix[i] == '/' || infix[i] == '^') && (stack[top] == '+' || stack[top] == '-'))

{

Push(infix[i]);

}

else if ((infix[i] == '^') && (stack[top] == '/' || stack[top] == '\*')){

Push(infix[i]);

}

else if ((infix[i] == '/' || infix[i] == '\*') && (stack[top] == '\*' || stack[top] == '/')){

postfix[idx] = Pop();

idx++;

Push(infix[i]);

}

else if ((infix[i] == '+' || infix[i] == '-') && (stack[top] == '+' || stack[top] == '-')){

postfix[idx] = Pop();

idx++;

Push(infix[i]);

}

else if((infix[i] == '\*' || infix[i] == '/') && ( stack[top] == '^')){

while (stack[top] != '+' && stack[top] != '-' && top != -1){

postfix[idx] = Pop();

idx++;

}

if (stack[top] == '+' || stack[top] == '-'){

postfix[idx] = Pop();

idx++;

Push(infix[i]);

}

else

Push(infix[i]);

}

else if ((infix[i] == '+' || infix[i] == '-') && (stack[top] == '/' || stack[top] == '\*' || stack[top] == '^'))

{

while (stack[top] != '+' && stack[top] != '-' && top != -1)

{

postfix[idx] = Pop();

idx++;

}

if (stack[top] == '+' || stack[top] == '-')

{

postfix[idx] = Pop();

idx++;

Push(infix[i]);

}

else

{

Push(infix[i]);

}

}

else if ((infix[i] == '(') && (stack[top] == '/' || stack[top] == '\*' || stack[top] == '^' || stack[top] == '+' || stack[top] == '-' || stack[top]=='('))

{

Push(infix[i]);

}

else if ((infix[i] == '+' || infix[i] == '-' || infix[i] == '/' || infix[i] == '\*' || infix[i] == '^') && (stack[top] == '('))

{

Push(infix[i]);

}

else if ((infix[i] == ')') && (stack[top] == '/' || stack[top] == '\*' || stack[top] == '^' || stack[top] == '+' || stack[top] == '-'))

{

while (stack[top] != '(')

{

postfix[idx] = Pop();

idx++;

}

Pop();

}

}

}

i++;

}

while (top != -1)

{

postfix[idx] = Pop();

idx++;

}

printf("Postfix expression is->\n");

int j = 0;

while (idx != j)

{

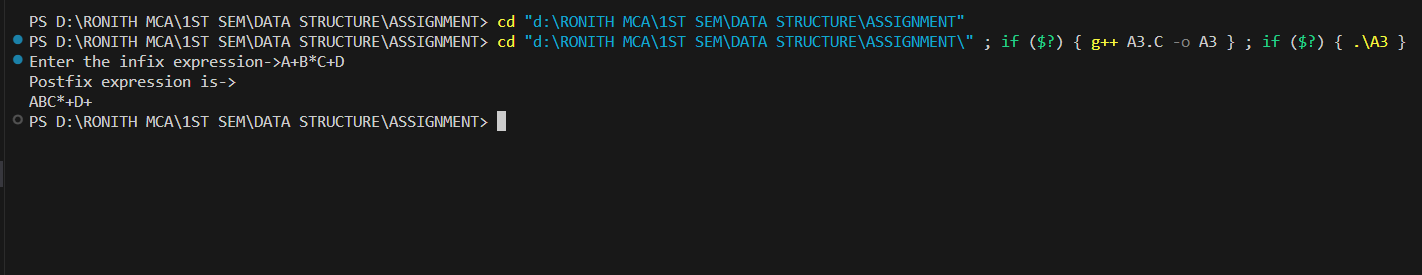
printf("%c", postfix[j]);

j++;

}

}

**Output :**

****

**Question 4:** Write a C program to convert an infix expression to its equivalent prefix expression.

**Source Code :**

#include<stdio.h>

#include<string.h>

char stack[100];

int top = -1;

int idx=0;

void Push(char item){

top += 1;

stack[top] = item;

}

char Peek(){

return stack[top];

}

char Pop(){

char item = stack[top];

top -= 1;

return item;

}

int main(){

char infix[100],infix\_rev[100],prefix\_rev[100],prefix[100];

printf("Enter infix expression->");

gets(infix);

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

if(infix[strlen(infix)-(i+1)]!='\0' && infix[strlen(infix)-(i+1)]!='\n' ){

infix\_rev[i]=infix[strlen(infix)-(i+1)];

}

}

for (int i = 0; i <= strlen(infix\_rev); i++){

if((infix\_rev[i]>='A' && infix\_rev[i]<='Z') || (infix\_rev[i]>='a' && infix\_rev[i]<='z')){

prefix\_rev[idx]=infix\_rev[i];

idx++;

}

else{

if(top==-1){

Push(infix\_rev[i]);

}

else{

if((infix\_rev[i]=='+' || infix\_rev[i]=='-') && (stack[top]=='+' || stack[top]=='-')){

Push(infix\_rev[i]);

}

else if((infix\_rev[i]=='\*' || infix\_rev[i]=='/') && (stack[top]=='\*' || stack[top]=='/')){

Push(infix\_rev[i]);

}

else if((infix\_rev[i]=='/' || infix\_rev[i]=='\*') && (stack[top]=='+' || stack[top]=='-')){

Push(infix\_rev[i]);

}

else if ((infix\_rev[i] == '+' || infix\_rev[i] == '-') && (stack[top] == '/' || stack[top] == '\*' || stack[top] == '^'))

{

while (stack[top] != '+' && stack[top] != '-' && top != -1 && stack[top]!=')'){

prefix\_rev[idx] = Pop();

idx++;

}

Push(infix\_rev[i]);

}

else if((infix\_rev[i]==')') && (stack[top]=='+' || stack[top]=='-' || stack[top]=='/' || stack[top]=='\*' || stack[top]=='^')){

Push(infix\_rev[i]);

}

else if((stack[top]==')') && (infix\_rev[i]=='+' || infix\_rev[i]=='-' || infix\_rev[i]=='/' || infix\_rev[i]=='\*' || infix\_rev[i]=='^')){

Push(infix\_rev[i]);

}

else if((infix\_rev[i]=='(') && (stack[top]=='+' || stack[top]=='-' || stack[top]=='/' || stack[top]=='\*' || stack[top]=='^')){

while (stack[top] != ')'){

prefix\_rev[idx]=Pop();

idx++;

}

Pop();

}

}

}

}

while(top!=-1){

prefix\_rev[idx]=Pop();

idx++;

}

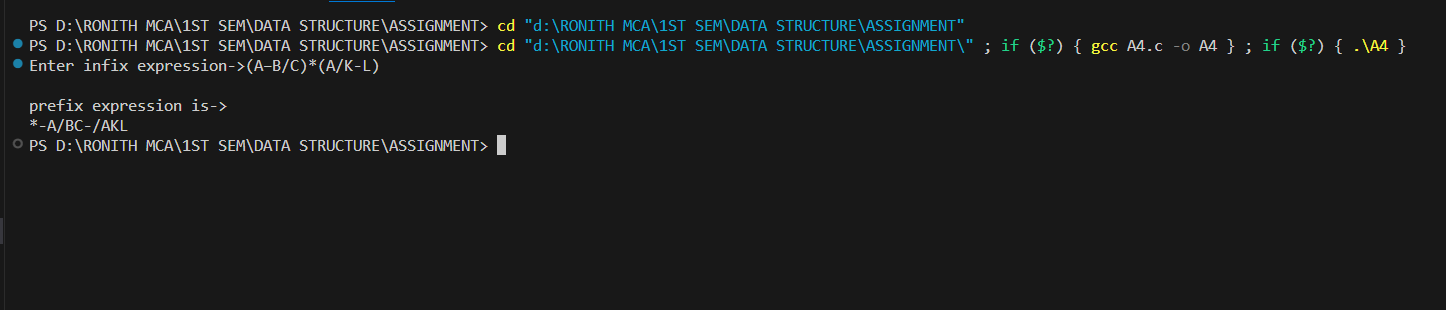
printf("\nprefix expression is->\n");

for (int i = idx-1; i >=0 ; i--)

printf("%c",prefix\_rev[i]);

}

**Output :**



**Question 5:** Write a C program to evaluate a postfix expression.

**Source Code :**

#include <stdio.h>

#include <string.h>

char stack[100];

int top = -1;

void Push(char item)

{

top += 1;

stack[top] = item;

}

char Peek()

{

return stack[top];

}

char Pop()

{

int item = stack[top];

top -= 1;

return item;

}

void display(){

int temp=top;

printf("\n stack is->");

while (temp!= -1)

{

printf("\t%d",stack[temp]);

temp--;

}

}

int calculate(int a,int b,char opretaor){

if(opretaor=='+'){

return b+a;

}

else if (opretaor=='-')

{

return b-a;

}

else if (opretaor=='\*')

{

return b\*a;

}

else if(opretaor=='^'){

int pow=1;

for (int i = 1; i <= a; i++)

{

pow=b\*pow;

}

return pow;

}

return b/a;

}

int main(){

char postfix[100];

printf("Enter the postfix expression->");

gets(postfix);

int i=0;

while (postfix[i] != '\0'){

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

int curElm=(int) postfix[i]-48;

Push(curElm);

}

else{

int a=Pop();

int b=Pop();

int sum=calculate(a,b,postfix[i]);

Push(sum);

}

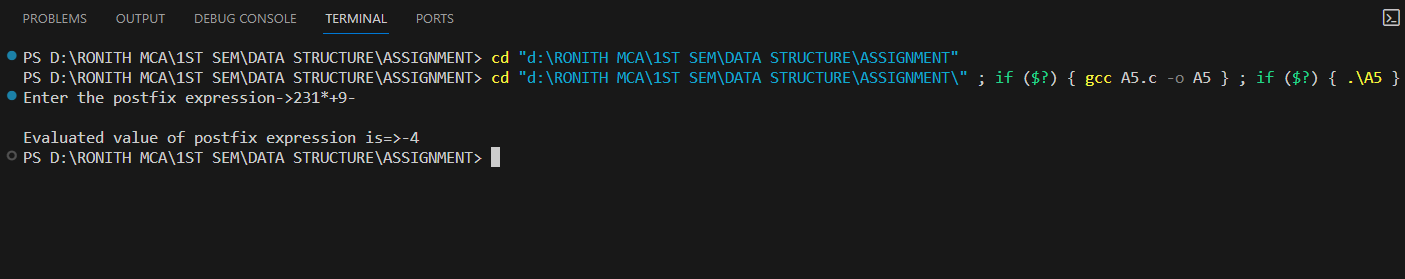
i++;

}

printf("\nEvaluated value of postfix expression is=>%d",Peek());

}

**Output :**

****

**Question 6:** Write a C program to evaluate a prefix expression.

**Source Code :**

#include <stdio.h>

#include <string.h>

char stack[100];

int top = -1;

void Push(char item)

{

top += 1;

stack[top] = item;

}

char Peek()

{

return stack[top];

}

char Pop()

{

int item = stack[top];

top -= 1;

return item;

}

void display(){

int temp=top;

printf("\n stack is->");

while (temp!= -1)

{

printf("\t%d",stack[temp]);

temp--;

}

}

int calculate(int a,int b,char opretaor){

if(opretaor=='+'){

return a+b;

}

else if (opretaor=='-')

{

return a-b;

}

else if (opretaor=='\*')

{

return a\*b;

}

else if(opretaor=='^'){

int pow=1;

for (int i = 1; i <= b; i++)

{

pow=a\*pow;

}

return pow;

}

return a/b;

}

int main(){

char prefix[100];

printf("Enter the prefix expression->");

gets(prefix);

for(int i=strlen(prefix)-1;i>=0;i--){

if(prefix[i]=='+' || prefix[i]=='-' || prefix[i]=='\*' || prefix[i]=='/' || prefix[i]=='^' ){

int a=Pop();

int b=Pop();

int sum=calculate(a,b,prefix[i]);

Push(sum);

}

else{

int curElm=(int) prefix[i]-48;

printf("\n%d",curElm);

Push(curElm);

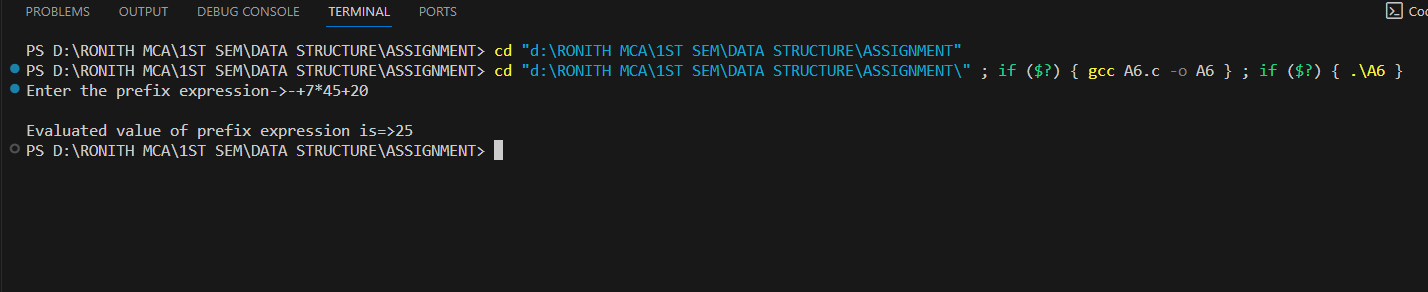
}

}

printf("\nEvaluated value of prefix expression is=>%d",Peek());

}

**Output :**



**Question 7:** Write a C program to generate fibonacci series using recursion.

**Source Code :**

#include<stdio.h>

int fibonacci(int n){

if(n==0) return 0;

else if(n==1) return 1;

else return fibonacci(n-1)+fibonacci(n-2);

}

int main(){

int n;

printf("enter the no of terms->");

scanf("%d",&n);

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

{

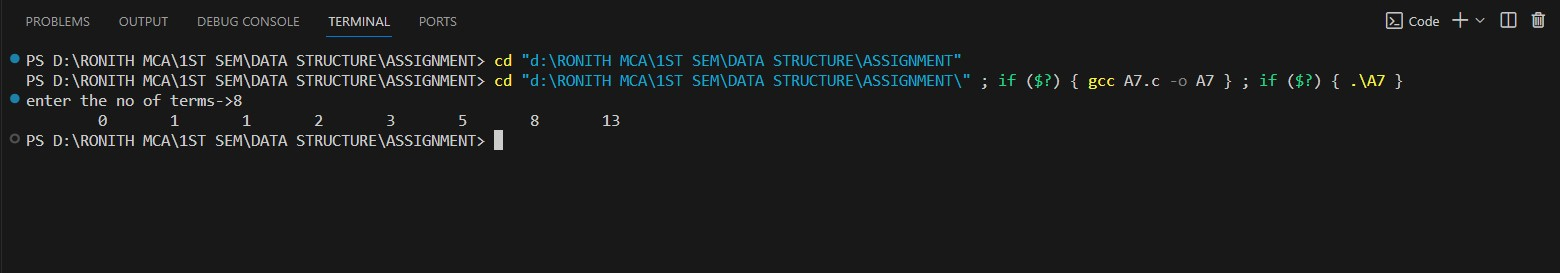
int f=fibonacci(i);

printf("\t%d",f);

}

}

**Output :**



**Question 8:** Write a C program to solve tower of hanoi problem using recursion.

**Source Code :**

#include<stdio.h>

int count=0;

void toh(int n,char s,char d,char a){

if(n>0){

toh(n-1,s,a,d);

printf("\nno of %d is placed from %c from %c",n,s,d);

toh(n-1,a,d,s);

}

}

int main(){

int n=2;

printf("\n No of rings are->%d",n);

char s='s';

char d='d';

char a='a';

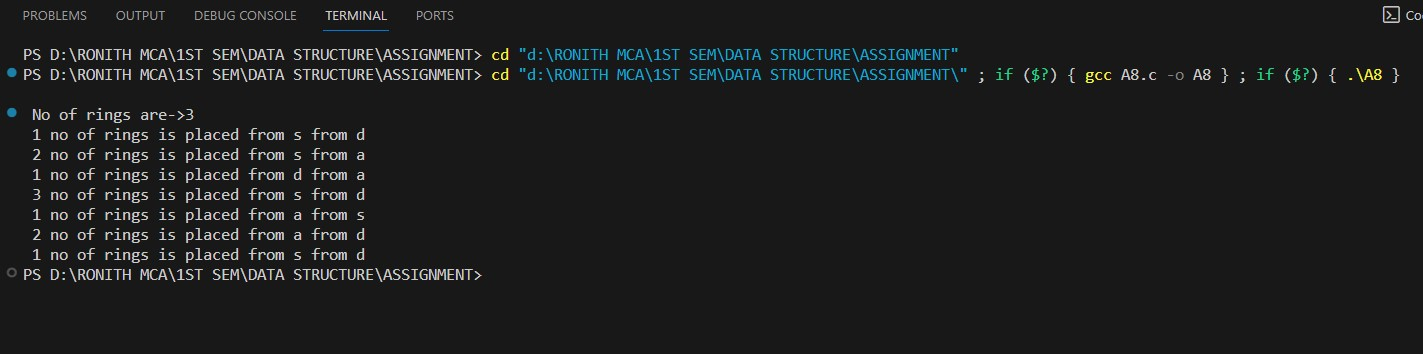
if(n>0){

toh(n,s,d,a);

}

}

**Output :**



**SEARCHING PROGRAMS**

**Question 1:**  Write a program to search an element in an array using linear search.

**Source Code :**

#include <stdio.h>

int main()

{

int array[100], search, c, number;

printf("Enter the number of elements in array\n");

scanf("%d",&number);

printf("Enter %d numbers\n", number);

for ( c = 0 ; c < number ; c++ )

scanf("%d",&array[c]);

printf("Enter the number to search\n");

scanf("%d",&search);

for ( c = 0 ; c < number ; c++ )

{

if ( array[c] == search ) /\* if required element found \*/

{

printf("%d is present at location %d.\n", search, c+1);

break;

}

}

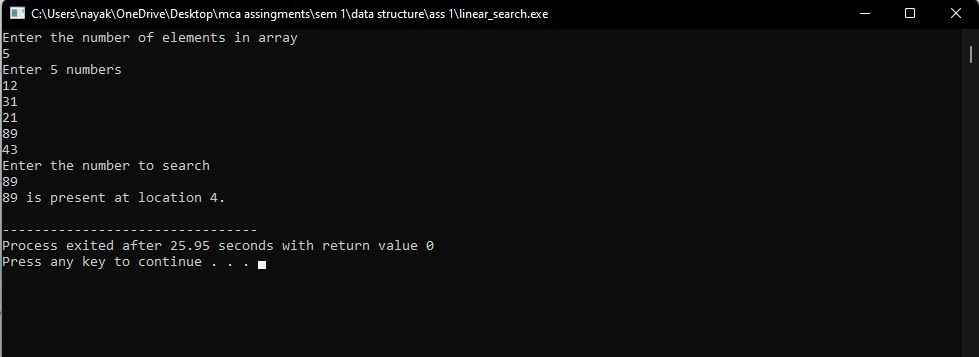
if ( c == number )

printf("%d is not present in array.\n", search);

return 0;

}

**Output :**



**Question 2:**  Write a program to search an element in an array using binary search.

**Source Code :**

#include <stdio.h>

int main()

{

int array[100], search, c, number;

printf("Enter the number of elements in array\n");

scanf("%d",&number);

printf("Enter %d numbers\n", number);

for ( c = 0 ; c < number ; c++ )

scanf("%d",&array[c]);

printf("Enter the number to search\n");

scanf("%d",&search);

for ( c = 0 ; c < number ; c++ )

{

if ( array[c] == search ) /\* if required element found \*/

{

printf("%d is present at location %d.\n", search, c+1);

break;

}

}

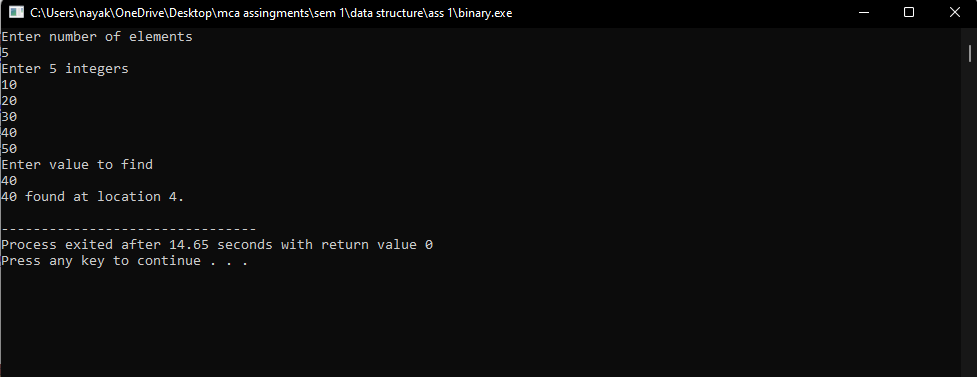
if ( c == number )

printf("%d is not present in array.\n", search);

return 0;

}

**Output :**

****