**IMPLEMENTATION OF LINEAR SEARCH AND BINARY SEARCH ALONG WITH TIME AND SPACE COMPLEXITY EVALUATION**

PROGRAM (LINEAR SEARCH):

#include <stdio.h>

void main(){

int n,m,found = 0,count = 0;

count++;

count++;

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

scanf("%d",&n);

count++;

int array[n];

printf("Enter the numbers: ");

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

count++;

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

count++;

}

count++;

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

scanf("%d",&m);

count++;

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

count++;

if (array[i] == m){

count++;

found = 1;

count++;

}

}

if (found == 1){

count++;

printf("The element is present\n");

}

else{

count++;

printf("The element is not found\n");

}

count++;

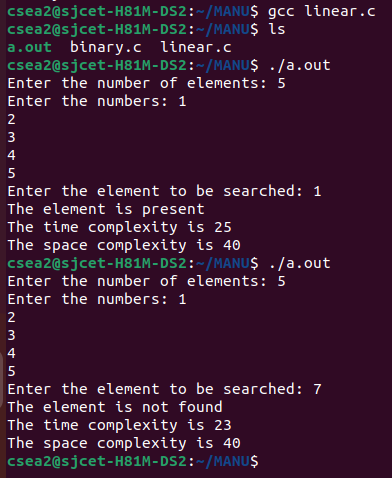
count++;

printf("The time complexity is %d\n",count);

printf("The space complexity is %d\n",20+(4\*n));

}

OUTPUT:



PROGRAM (BINARY SEARCH):

#include <stdio.h>

void main(){

int n,m,result,temp,key = 0,found=0;

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

scanf("%d",&n);

int array[n],left = 0,right = n-1,middle;

printf("Enter the elements: ");

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

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

}

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

if (array[i] = array[i+1]){

temp = array[i];

array[i] = array[i+1];

array[i+1] = temp;

}

}

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

scanf("%d",&m);

while (left <= right){

middle = (left+right)/2;

if (m == array[middle]){

printf("The element is found\n");

found = 1;

break;

}

else if (m < array[middle]){

left = 0;

right = middle-1;

}

else if (m > array[middle]){

left = middle+1;

right = n-1;

}

}

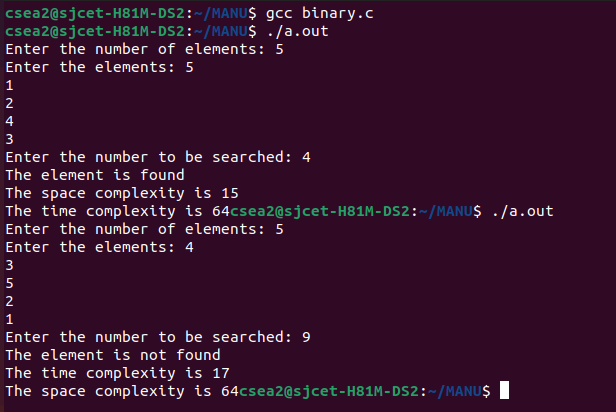
if (found == 0){

printf("The element is not found\n");

}

}

OUTPUT:



**SPARSE MATRIX REPRESENTATION USING ARRAYS**

PROGRAM:

#include <stdio.h>

void main(){

int row,column,matrix[20][20],count = 0;

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

scanf("%d%d",&row,&column);

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

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

for (int j = 0; j < column; j ++){

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

}

}

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

for (int j = 0; j < column; j ++){

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

count++;

}

}

}

printf("The sparse matrix: \n");

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

for (int j = 0; j < column; j ++){

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

}

printf("\n");

}

int tuple[count+1][3];

tuple[0][0] = row;

tuple[0][1] = column;

tuple[0][2] = count;

int a = 1;

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

for (int j = 0; j < 3; j++){

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

tuple[a][0] = i;

tuple[a][1] = j;

tuple[a][2] = matrix[i][j];

a++;

}

}

}

printf("The tuple matrix: \n");

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

for (int j = 0; j < 3; j ++){

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

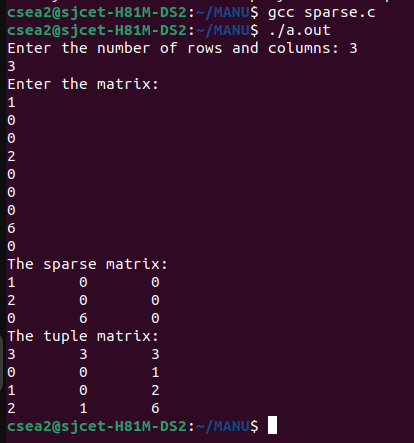
}

printf("\n");

}

}

OUTPUT:



**POLYNOMIAL REPRESENTATION USING STRUCTURES**

PROGRAM:

#include <stdio.h>

void main(){

int n;

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

scanf("%d",&n);

struct polynomial{

int coefficient;

int exponent;

}a[n];

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

printf("Enter the coefficient of term %d: \n",i+1);

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

printf("Enter the exponent of term %d: \n",i+1);

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

}

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

if (a[i].coefficient < a[i+1].coefficient){

}

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

printf("x");

printf("^%d",a[i].exponent);

if (n > 1){

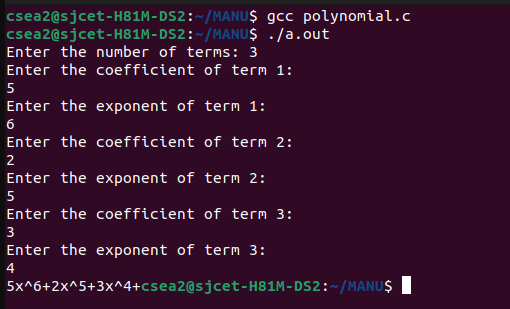
printf("+");

}

}

}

OUTPUT:



**IMPLEMENTATION OF STACK AND QUEUE USING ARRAYS**

PROGRAM (STACK):

#include <stdio.h>

int top = -1;

int stack[10];

int push(int n, int max);

int display(int max);

int pop();

int peek();

void main(){

int n,max, k = 0;

printf("Enter the limit: ");

scanf("%d",&max);

while (k == 0){

printf(" 1.Push\n 2.Pop\n 3.Peek\n 4.Display\n 5.Exit\n Enter a choice: ");

scanf("%d",&n);

if (n == 1){

int num;

top = top + 1;

printf("Enter the number: ");

scanf("%d",&num);

push(num,max);

}

else if (n == 2) {

pop();

}

else if (n == 4){

display(max);

}

else if ( n == 3){

peek();

}

}

}

int peek(){

if (top == -1){

printf("\nUNDERFLOW\n");

}

else {

printf("The top-most element is: %d",stack[top]);

}

}

int push(int n, int max){

if (top >= max - 1 ){

printf("\nOVERFLOW\n");

}

else {

stack[top] = n;

}

}

int pop(){

if (top == -1){

printf("\nUNDERFLOW\n");

}

else {

top--;

}

}

int display(int max){

if (top == -1){

printf("\nThe stack is EMPTY \n");

}

printf("\n");

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

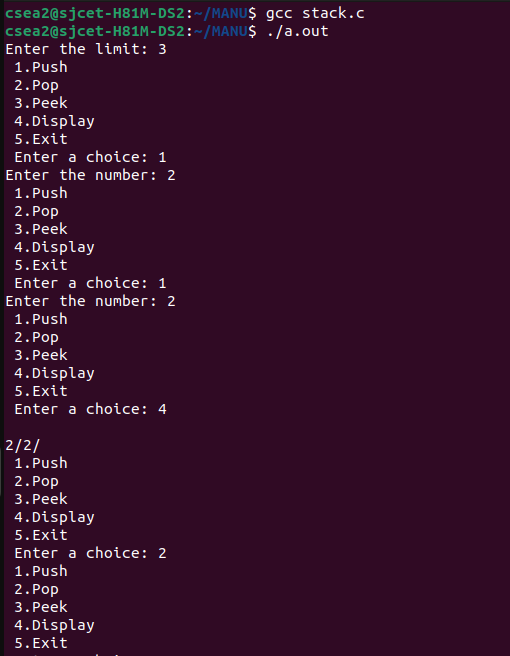
printf("%d/",stack[i]);

}

printf("\n");

}

OUTPUT:



PROGRAM (QUEUE):

#include <stdio.h>

int rear = -1, front = -1,queue[20];

int enqueue(int item,int max);

int dequeue();

int display();

void main(){

int n,choice,num,k = 1;

printf("Enter the maximum size of the queue: ");

scanf("%d",&num);

while (k != 0){

printf("\n 1. Enqueue\n 2. Dequeue\n 3. Display\n 4. Exit\n Enter a choice: ");

scanf("%d",&choice);

if (choice == 1){

printf("Enqueue : ");

printf("Enter a number: ");

scanf("%d",&n);

enqueue(n,num);

}

else if (choice == 3){

display();

}

else if (choice == 2){

dequeue();

}

else if(choice == 4){

break;

}

}

}

int dequeue(){

if (rear <= front){

printf("\nThe queue is EMPTY.\n");

rear = -1;

front = -1;

}

else {

front++;

printf("The dequeued element is %d",queue[front-1]);

}

}

int display(){

printf("The queue is: \n");

for (int i = front; i < rear+1; i++){

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

}

}

int enqueue(int item, int max){

if (front == - 1 && rear == -1){

rear = 0;

front = 0;

queue[rear] = item;

}

else if(rear == max - 1){

printf("\nOVERFLOW\n");

}

else {

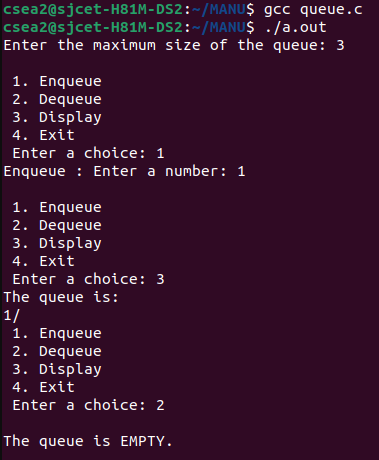
rear++;

queue[rear] = item;

}

}

OUTPUT:



**IMPLEMENTATION OF CIRCULAR QUEUE USING ARRAYS**

PROGRAM:

#include <stdio.h>

int rear = -1, front = -1, queue[20], max;

int enqueue(int item);

int dequeue();

int display();

void main() {

int choice, num, k = 1;

printf("Enter the maximum size of the queue: ");

scanf("%d", &max);

while (k != 0) {

printf("\n1.Enqueue\n2.Dequeue\n3.Display\n4.Exit\nEnter a choice: ");

scanf("%d", &choice);

if (choice == 1) {

printf("Enter a number: ");

scanf("%d", &num);

enqueue(num);

}

else if (choice == 3) {

display();

}

else if (choice == 2) {

dequeue();

}

else if (choice == 4) {

break;

}

}

}

int enqueue(int item) {

if ((front == 0 && rear == max - 1) || (rear == front - 1 && front != -1)) {

printf("\nOVERFLOW\n");

} else {

if (front == -1) {

front = 0;

}

rear = (rear + 1) % max;

queue[rear] = item;

}

}

int dequeue() {

if (front == -1) {

printf("\nThe queue is EMPTY.\n");

} else {

printf("The dequeued element is %d", queue[front]);

if (front == rear) {

front = -1;

rear = -1;

} else {

front = (front + 1) % max;

}

}

}

int display() {

printf("The queue is: \n");

int i = front;

while (i != rear) {

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

i = (i + 1) % max;

}

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

printf("\nRear: %d",rear);

printf("\nFront: %d",front);

}

OUTPUT:

**CONVERTION OF EXPRESSION FROM ONE NOTATION TO ANOTHER**

PROGRAM:

OUTPUT:

**IMPLEMENTATION OF BUBBLE SORT AND SELECTION SORT ALONG WITH SPACE AND TIME COMPLEXITY EVALUATION**

PROGRAM (BUBBLE SORT):

OUTPUT:

PROGRAM (SELECTION SORT):

OUTPUT: