K.GEETIKA
AP19110010123
CSE-G

15. Write a C Program to Sort the Array in an Ascending Order using Bubble sort.

PROBLEM STATEMENT In this problem we aim to understand the usage of loops and arrays in c language by considering a maths problem. It requires input from the user such as:
I, j, array ,n
Once input data is collected and sorted, the elements are stored in an array and sorted in ascending order by bubble sort.

ALGORITHM
START DEFINE VARIABLES: array, n, I, j.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the elements in the array, So sorting is done in ascending order by using bubble sort.
DISPLAY: Ascending ordered array.
STOP

PROGRAM
```
 #include<stdio.h>
int main(){
int array[50], n, i, j, temp;

printf("ENTER THE NUMBER OF ELEMENTS\n");
scanf("%d", &n);
printf("ENTER %d INTEGERS\n", n);

for(i = 0; i < n; i++)
scanf("%d", &array[i]);
for (i = 0 ; i < ( n - 1 ); i++){
for (j= 0 ; j < n - i - 1; j++){
if(array[j] > array[j+1]){
temp=array[j];
array[j]   = array[j+1];
array[j+1] = temp;
}
}
}

printf("SORTED LIST IN ASCENDING ORDER:\n");
for ( i = 0 ; i < n ; i++ )
printf("%d\n", array[i]);
return 0;
}
```

OUTPUT

```
 ENTER THE NUMBER OF ELEMENTS
4
ENTER 4 INTEGERS
1
3
5
2
SORTED LIST IN ASCENDING ORDER:
1
2
3
5
```

16. Write a C Program to sort an array in descending order using Selection sort.

PROBLEM STATEMENT In this problem we aim to understand the usage of loops and arrays in c language by considering a maths problem. It requires input from the user such as:
I, j, a ,n
Once input data is collected and sorted, the elements are stored in an array and sorted in descending order.

ALGORITHM
START DEFINE VARIABLES: I, j, a, n
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the elements in the array, So sorting is done in descending order by selection sort.
DISPLAY: Descended Array.
STOP

PROGRAM

```
 #include <stdio.h>
void main ()
{

int number[30];

int i, j, a, n;
printf("ENTER THE VALUE OF N\n");
scanf("%d", &n);

printf("ENTER THE NUMBER \n");
for (i = 0; i < n; ++i)
scanf("%d", &number[i]);
for (i = 0; i < n; ++i)
{
for (j = i + 1; j < n; ++j)
{
if (number[i] < number[j])
{
   a = number[i];
number[i] = number[j];
number[j] = a;
}
}
}
printf("THE NUMBERS ARRANGED IN DESCENDING ORDER ARE GIVEN BELOW\n");
for (i = 0; i < n; ++i)
{
printf("%d\n", number[i]);
}
}
```

OUTPUT

```
 ENTER THE VALUE OF N
5
ENTER THE NUMBER
10
1
4
7
3
THE NUMBERS ARRANGED IN DESCENDING ORDER ARE GIVEN BELOW
10
7
4
3
1
```

17. Write a C Program to sort an array in ascending order using Insertion sort.

PROBLEM STATEMENT In this problem we aim to understand the usage of loops and arrays in c language by considering a maths problem. It requires input from the user such as:

I, j, number

Once input data is collected and sorted using insertion sort, the elements are stored in an array and sorted in ascending order.

ALGORITHM

START DEFINE VARIABLES: I, j, number.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the elements in the array, So sorting is performed in ascending order by insertion sort.
DISPLAY: Ascended array.
STOP

PROGRAM

```
#include<stdio.h>
int main()
{
int i, j, count, temp, number[25];

printf("HOW MANY ELEMENTS YOUR GOING TO ENTER?: ");
scanf("%d",&count);

printf("ENTER %d ELEMENTS: ", count);

for(i=0;i<count;i++)
scanf("%d",&number[i]);

for(i=1;i<count;i++)
{
temp=number[i];
j=i-1;
while((temp<number[j])&&(j>=0))
{
number[j+1]=number[j];
j=j-1;
}
number[j+1]=temp;
}
```

OUTPUT

```
 HOW MANY ELEMENTS YOUR GOING TO ENTER?: 4
ENTER 4 ELEMENTS: 10
20
```

```
50
70
ORDER OF SORTED ELEMENTS:  10 20 50 70
```

18. Write a C Program to sort an array in descending order using Heapsort.

PROBLEM STATEMENT In this problem we aim to understand the usage of loops and arrays in c language by considering a maths problem. It requires input from the user such as:
arr_sort
Once input data is collected and sorted, the elements are stored in an array and sorted in descending order.

ALGORITHM
START DEFINE VARIABLES: arr_sort.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the elements to rearrange the array in descending by heapsort.
DISPLAY:

STOP

PROGRAM
```
#include<stdio.h>
#include<conio.h>
#define MAX_SIZE 5

void heap_sort();

void heap_adjust(int, int);
int arr_sort[MAX_SIZE], t, a;


int main() {
int i;

printf("\nENTER %d ELEMENTS FOR SORTING\n", MAX_SIZE);
for (i = 0; i < MAX_SIZE; i++)
scanf("%d", &arr_sort[i]);



heap_sort();

printf("\n\nSORTED DATA :");
for (i = MAX_SIZE-1; i >= 0; i--) {
printf("\t%d", arr_sort[i]);
}
}

void heap_sort() {
for (int i = MAX_SIZE / 2 - 1; i >= 0; i--)
heap_adjust(MAX_SIZE, i);

for (int i = MAX_SIZE - 1; i >= 0; i--) {
   t = arr_sort[0];
   arr_sort[0] = arr_sort[i];
   arr_sort[i] = t;
   heap_adjust(i, 0);


}
}


void heap_adjust(int n, int i) {
int large = i, left = 2 * i + 1, right = 2 * i + 2;

if (left < n && arr_sort[left] > arr_sort[large])
large = left;

if (right < n && arr_sort[right] > arr_sort[large])
   large = right;


if (large != i) {

   t = arr_sort[i];
   arr_sort[i] = arr_sort[large];
   arr_sort[large] = t;
   heap_adjust(n, large);
}
}
```
OUTPUT
```
 ENTER 5 ELEMENTS FOR SORTING
10
130
56
```

```
82
70


SORTED DATA :   130     82      70      56      10
```
19. Write a C program that takes an array and returns a new array with unique elements of the first array.

PROBLEM STATEMENT In this problem we aim to the usage of loops and arrays in c language. By considering an operation in computer language. It requires Input from user such as:

Size, I, j, count

Once the details are collected and stored, then the unique element is created with a new array.

ALGORITHM

START DEFINE VARIABLE: size, I, j, count.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter an array so that it generates a new array.
DISPLAY: Unique element
STOP

PROGRAM

```
  #include <stdio.h>
#define MAX_SIZE 100

int main()
{
int arr[MAX_SIZE], freq[MAX_SIZE];
int size, i, j, count;

printf("ENTER SIZE OF THE ARRAY: ");
scanf("%d", &size);
printf("ENTER THE ELEMENTS IN ARRAY: ");
for(i=0; i<size; i++)
{
scanf("%d", &arr[i]);
freq[i] = -1;
}

for(i=0; i<size; i++)
{
count = 1;
for(j=i+1; j<size; j++)
{
if(arr[i] == arr[j])
{
count++;
freq[j] = 0;
}
}
if(freq[i] != 0)
{
freq[i] = count;
}
}
printf("\nUNIQUE ELEMENT IN THE ARRAY ARE: ");
for(i=0; i<size; i++)
{
if(freq[i] == 1)
{
printf("%d ", arr[i]);
}
}

return 0;
}
```

OUTPUT

```
 ENTER SIZE OF THE ARRAY: 10
ENTER THE ELEMENTS IN ARRAY: 1 2 3 5 1 7 20 2 12 10

UNIQUE ELEMENT IN THE ARRAY ARE: 3 5 7 20 12 10
```

20. Write a C Program to input two matrices of 5×5 add them and output the resultant matrix.

PROBLEM STATEMENT In this problem we aim to understand the usage of data types, operators and conditions in c language by considering a maths problem. It is required to give input such as:
a[ ] [ ] , b[ ] [ ]

Once these details are given, The sum of the matrix will be displayed.

ALGORITHM
 START DEFINE VALUE: r, c, a, b, sum, I, j.

INPUT: Read the information from keyboard
COMPUTATION: Enter the information, to generate the matrix.
DISPLAY: Resultant Matrix
STOP

PROGRAM

```
#include <stdio.h>
int main() {
int r, c, a[100][100], b[100][100], sum[100][100], i, j;
printf("ENTER THE NUMBER OF ROWS [BETWEEN 1 AND 100]: ");
scanf("%d", &r);
printf("ENTER THE NUMBER OF COLUMNS [BETWEEN 1 AND 100]: ");
scanf("%d", &c);
printf("\nENTER ELEMENTS OF 1ST MATRIX:\n");
for (i = 0; i < r; ++i)
for (j = 0; j < c; ++j)
{
printf("ENTER ELEMENT A%d%d: ", i + 1, j + 1);
scanf("%d", &a[i][j]);
}

printf("ENTER ELEMENTS OF 2nd MATRIX:\n");
for (i = 0; i < r; ++i)
for (j = 0; j < c; ++j) {
printf("ENTER ELEMENT A%d%d: ", i + 1, j + 1);
scanf("%d", &b[i][j]);
}

for (i = 0; i < r; ++i)
for (j = 0; j < c; ++j) {
sum[i][j] = a[i][j] + b[i][j];
}

printf("\nSum of two matrices: \n");
for (i = 0; i < r; ++i)
for (j = 0; j < c; ++j) {
printf("%d   ", sum[i][j]);
if (j == c - 1)
{
printf("\n\n");
}
}
return 0;
}
```
OUTPUT
```
 ENTER THE NUMBER OF ROWS [BETWEEN 1 AND 100]: 5
ENTER THE NUMBER OF COLUMNS [BETWEEN 1 AND 100]: 5

ENTER ELEMENTS OF 1ST MATRIX:
ENTER ELEMENT A11: 1
ENTER ELEMENT A12: 1
ENTER ELEMENT A13: 1
ENTER ELEMENT A14: 1
ENTER ELEMENT A15: 1
ENTER ELEMENT A21: 1
ENTER ELEMENT A22: 1
ENTER ELEMENT A23: 1
ENTER ELEMENT A24: 1
ENTER ELEMENT A25: 1
ENTER ELEMENT A31: 1
ENTER ELEMENT A32: 1
ENTER ELEMENT A33: 1
ENTER ELEMENT A34: 1
ENTER ELEMENT A35: 1
ENTER ELEMENT A41: 1
ENTER ELEMENT A42: 1
ENTER ELEMENT A43: 1
ENTER ELEMENT A44: 1
ENTER ELEMENT A45: 1
ENTER ELEMENT A51: 1
ENTER ELEMENT A52: 1
ENTER ELEMENT A53: 1
ENTER ELEMENT A54: 1
ENTER ELEMENT A55: 1
 ENTER ELEMENTS OF 2nd MATRIX:
ENTER ELEMENT A11: 1
ENTER ELEMENT A12: 1
ENTER ELEMENT A13: 1
ENTER ELEMENT A14: 1
ENTER ELEMENT A15: 1
ENTER ELEMENT A21: 1
ENTER ELEMENT A22: 1
ENTER ELEMENT A23: 1
ENTER ELEMENT A24: 1
ENTER ELEMENT A25: 1
ENTER ELEMENT A31: 1
ENTER ELEMENT A32: 1
ENTER ELEMENT A33: 1
ENTER ELEMENT A34: 1
ENTER ELEMENT A35: 1
ENTER ELEMENT A41: 1
ENTER ELEMENT A42: 1
ENTER ELEMENT A43: 1
ENTER ELEMENT A44: 1
ENTER ELEMENT A45: 1
ENTER ELEMENT A51: 1
ENTER ELEMENT A52: 1
ENTER ELEMENT A53: 1
ENTER ELEMENT A54: 1
ENTER ELEMENT A55: 1
  Sum of two matrices:
2   2   2   2   2

2   2   2   2   2

2   2   2   2   2

2   2   2   2   2

2   2   2   2   2
```

21. Write a C Program to input two matrices of 5×5 multiply them and output the resultant matrix.

PROBLEM STATEMENT In this problem we aim to understand the usage of data types, operators and conditions in c language by considering a maths problem. It is required to give input such as:
a[ ] [ ] , b[ ] [ ]

Once these details are given, The multiplier of the matrix will be displayed.

ALGORITHM
START DEFINE VALUE: a, b, c, I, j, k.
INPUT: Read the information from keyboard
COMPUTATION: Enter the information, to generate the product of the matrix.
DISPLAY: Resultant Matrix
STOP

PROGRAM

```
 #include<stdio.h>
void main()
{
int a[10][10],b[10][10],c[10][10],i,j,k;
for(i=0;i<5;i++)
{
for(j=0;j<5;j++)
{
printf("ENTER THE VALUE OF A[%d][%d]",i,j);
scanf("%d",&a[i][j]);
}
}
for(i=0;i<5;i++)
{
for(j=0;j<5;j++)
{
printf("ENTER THE VALUE OF B[%d][%d]",i,j);
scanf("%d",&b[i][j]);
}
}
for(i=0;i<5;i++)
{
for(j=0;j<5;j++)
{
c[i][j]=0;
for(k=0;k<5;k++)
{
c[i][j]=c[i][j]+a[i][k]*b[k][j];
}
}
}
printf("RESULTANT MATRIX IS \n");
for(i=0;i<5;i++)
{
for(j=0;j<5;j++)
{
printf("%d ",c[i][j]);
}
printf("\n");
}
}
```

OUTPUT

```
 ENTER THE VALUE OF A[0][0]1
ENTER THE VALUE OF A[0][1]1
ENTER THE VALUE OF A[0][2]1
ENTER THE VALUE OF A[0][3]1
ENTER THE VALUE OF A[0][4]1
ENTER THE VALUE OF A[1][0]1
ENTER THE VALUE OF A[1][1]1
ENTER THE VALUE OF A[1][2]1
ENTER THE VALUE OF A[1][3]1
ENTER THE VALUE OF A[1][4]1
ENTER THE VALUE OF A[2][0]1
ENTER THE VALUE OF A[2][1]1
ENTER THE VALUE OF A[2][2]1
ENTER THE VALUE OF A[2][3]1
ENTER THE VALUE OF A[2][4]1
ENTER THE VALUE OF A[3][0]1
ENTER THE VALUE OF A[3][1]1
ENTER THE VALUE OF A[3][2]1
ENTER THE VALUE OF A[3][3]1
ENTER THE VALUE OF A[3][4]1
ENTER THE VALUE OF A[4][0]1
ENTER THE VALUE OF A[4][1]1
ENTER THE VALUE OF A[4][2]1
ENTER THE VALUE OF A[4][3]1
ENTER THE VALUE OF A[4][4]1
ENTER THE VALUE OF B[0][0]1
ENTER THE VALUE OF B[0][1]1
ENTER THE VALUE OF B[0][2]1
ENTER THE VALUE OF B[0][3]1
ENTER THE VALUE OF B[0][4]1
ENTER THE VALUE OF B[1][0]1
ENTER THE VALUE OF B[1][1]1
ENTER THE VALUE OF B[1][2]1
ENTER THE VALUE OF B[1][3]1
ENTER THE VALUE OF B[1][4]1
ENTER THE VALUE OF B[2][0]1
ENTER THE VALUE OF B[2][1]1
ENTER THE VALUE OF B[2][2]1
ENTER THE VALUE OF B[2][3]1
ENTER THE VALUE OF B[2][4]1
ENTER THE VALUE OF B[3][0]1
ENTER THE VALUE OF B[3][1]1
ENTER THE VALUE OF B[3][2]1
ENTER THE VALUE OF B[3][3]1
ENTER THE VALUE OF B[3][4]1
ENTER THE VALUE OF B[4][0]1
ENTER THE VALUE OF B[4][1]1
ENTER THE VALUE OF B[4][2]1
ENTER THE VALUE OF B[4][3]1
ENTER THE VALUE OF B[4][4]1
  RESULTANT MATRIX IS
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
```

22. Write a C Program to find the sum of natural numbers using function.

PROBLEM STATEMENT In this problem we aim to find the sum of natural numbers using functions. It is required to give input such as:
num

Once input data is collected and stored, the sum is printed according to the value given by the user.

ALGORITHM

```
START DEFINE VARIABLES: num.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the value to create natural numbers by functions.
DISPLAY: Sum.
STOP
```

PROGRAM

```
 #include <stdio.h>
int addNumbers(int n);
int main()
{
int num;
printf("ENTER A +VE INTEGER: ");
scanf("%d", &num);
printf("SUM = %d", addNumbers(num));
return 0;
}

int addNumbers(int n)
```

```
{
if (n != 0)
return n + addNumbers(n - 1);
else
return n;
}
```

OUTPUT

```
 ENTER A +VE INTEGER: 7
SUM = 28
```

23. Write a C Program to find factorial of number using recursion.

PROBLEM STATEMENT In this problem we aim to calculate the factorial value of numbers using recursion. It is required to give input such as:
n

Once the data is collected and stored, the factorial value of the numbers is computed and the output is printed

ALGORITHM

```
 START DEFINE VARIABLES: n.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the number to create a factorial by recursion.
DISPLAY: Factorial
STOP
```

PROGRAM

```
 #include<stdio.h>
long int multiplyNumbers(int n);
int main() {
int n;
printf("A +VE INTEGER TO BE ENTERED: ");
scanf("%d",&n);
printf("FACTORIAL OF %d = %ld", n, multiplyNumbers(n));
return 0;
}

long int multiplyNumbers(int n) {
if (n>=1)
return n*multiplyNumbers(n-1);
else
return 1;
}
```

OUTPUT

```
 A +VE INTEGER TO BE ENTERED: 7
FACTORIAL OF 7 = 5040
```

24. Write a C Program to generate the Fibonacci series.

PROBLEM STATEMENT In this problem we aim to calculate the Fibonacci sequence. It is required to give input such as:
n

Once the input data is collected and stored, the Fibonacci sequence is computed and the output is printed.

ALGORITHM

```
 START DEFINE VARIABLE: n
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the value of n to create Fibonacci series.
DISPLAY: Fibonacci number
STOP
```

PROGRAM

```
 #include <stdio.h>
int fib(int n)
{
if(n<=1)
return n;
return(fib(n-1)+fib(n-2));
}
int main()
{
int n=9;
printf("THE FIB SERIES IS %d",fib(n));
}
```

OUTPUT

```
THE FIB SERIES IS 0, 1, 1, 2, 3, 5, 8, 13, 21, 34
```

25. Write a C Program using structure for entering details of the five students as name, admission number, Date of birth, department and display all the details.

PROBLEM STATEMENT In this problem we aim to understand the usage of loops and structures in c language by considering a real-life scenario. It is required to give information such as :

roll_num , student=stu , name, fee, DOB
Once these details are given, the information of the student is generated.

ALGORITHM
START DEFINE VARIABLES: roll_num , stu ,name, fee, DOB.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the total number of students so that the program will create that many students. So that their details could be entered.
DISPLAY: Details of the 5 students.
STOP
PROGRAM
```
#include<stdio.h>
int main()
{
struct student
{
int roll_num;
char name[60];
int fee;
char DOB[80];
};
struct student stu[90];
int a,b;
printf("\n ENTER THE NUMBER OF STUDENTS");
scanf("%d",&a);
for(b=0;b<a;b++)
{
printf("\n ENTER THE ROLL NUMBER");
scanf("%d",&stu[b].roll_num);
printf("\n ENTER THE NAME");
scanf("%s",&stu[b].name);
printf("\n ENTER THE FEE");
scanf("%d",&stu[b].fee);
printf("\n ENTER THE DOB");
scanf("%s",&stu[b].DOB);
}
for(b=0;b<a;b++)
{
printf("\n DETAILS OF THE STUDENT YOU HAVE ASKED FOR %d",b+1);
printf("\n ROLL NO = %d",stu[b].roll_num);
printf("\n NAME = %s",stu[b].name);
printf("\n FEE = %d",stu[b].fee);
printf("\n DOB = %s",stu[b].DOB);
}
}
```

OUTPUT

```
 DETAILS OF THE STUDENT YOU HAVE ASKED FOR 1
 ROLL NO = 20
 NAME = GEETIKA
 FEE = 9000
 DOB = 21/11/2001
 DETAILS OF THE STUDENT YOU HAVE ASKED FOR 2
 ROLL NO = 30
 NAME = HARSHITHA
 FEE = 6000
 DOB = 24/02/2002
 DETAILS OF THE STUDENT YOU HAVE ASKED FOR 3
 ROLL NO = 14
 NAME = KUSUMA
 FEE = 8000
 DOB = 23/11/2002
 DETAILS OF THE STUDENT YOU HAVE ASKED FOR 4
 ROLL NO = 50
 NAME = VAISHNAVI
 FEE = 8500
 DOB = 21/03/2002
 DETAILS OF THE STUDENT YOU HAVE ASKED FOR 5
 ROLL NO = 42
 NAME = SAKITHA
 FEE = 6200
 DOB = 12/06/2002
```

26. Write a C program to find length of string using pointers.

PROBLEM STATEMENT In this problem we aim to understand the usage of loops, pointers and operators by considering an example from computer language. It is required to give information such as:
string
Once this detail is entered, The program will run to generate the length of the string.

ALGORITHM
 START DEFINE VARIABLES: string.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the string so that it generates the length of the string.
DISPLAY: Length of the string.
STOP

PROGRAM

```
#include<stdio.h>
int stringlength(char*);
void main()
{
char string[20];
int length;
printf("\n THE STRING TO BE ENTERED : ");
gets(string);
length = stringlength(string);
printf("THE STRING LENGTH TO BE ENTERED %s IS : %d", string, length);
}
int stringlength(char*p)
{
int count = 0;
while (*p != '\0') {
count++;
p++;
}
return count;
}
```

OUTPUT

```
THE STRING TO BE ENTERED: GEETIKA
THE STRING LENGTH TO BE ENTERED GEETIKA IS : 6
```

27.Write a C program to copy one string to another using pointers.

PROBLEM STATEMENT In this problem we aim to understand the usage of loops, pointers and operators by considering an example from computer language. It is required to give information such as:
str, copy_str,*pstr,*pcopy_str
Once these details are entered, The program will create a copy of the string.

ALGORITHM

```
START DEFINE VARIABLES: str, copy_str,*pstr,*pcopy_str.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the string so that it could generate the string by pointers.
DISPLAY: Copied String.
STOP
```

PROGRAM

```
#include<stdio.h>
```

```
#define MAX 100
int main()
{
char str[MAX],copy_str[MAX];
char*pstr,*pcopy_str;
int i=0;
pstr=str;
pcopy_str=copy_str;
printf("\n THE STRING TO BE ENTERED");
gets(str);
while(*pstr!='\0')
{
*pcopy_str=*pstr;
 pstr++,pcopy_str++;
}
*pcopy_str='\0';
printf("\n THE COPIED STRING IS");
pcopy_str= copy_str;
while(*pcopy_str!='\0')
{
  printf("%c",*pcopy_str);
pcopy_str++;
}
}
```

OUTPUT

```
 THE STRING TO BE ENTERED VIJAYAWADA
 THE COPIED STRING IS VIJAYAWADA
```

28. Write a C program to compare two strings using pointers.

PROBLEM STATEMENT In this problem we aim to understand the usage of loops, pointers and operators by considering an example from computer language. It is required to give information such as:
string1, string2,*str1,*str2,i
Once these details are entered, The program will create strings(user input)and then compare the string.

ALGORITHM

START DISPLAY VARIABLES: string1, string2, *str1, *str2.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the information in string 1 and string 2 to see whether they are equal or not.
DISPLAY: Whether the 2 strings are equal or not.
STOP

PROGRAM

```
#include<stdio.h>
#define MAX 100
int main()
{
char string1[MAX],string2[MAX],*str1,*str2;
int i,equal = 0;
printf("THE STRING TO BE ENTERED: ");
scanf("%s",string1);
printf("THE 2ND STRING TO BE ENTERED: ");
scanf("%s",string2);
str1 = string1;
str2 = string2;
while(*str1 == *str2)
{

if ( *str1 == '\0' || *str2 == '\0' )
break;
str1++;
str2++;
}
if( *str1 == '\0' && *str2 == '\0' )
printf("\n\nTHE STRINGS ARE SAME.");
else
printf("\n\nTHE STRINGS ARE NOT SAME.");
}
```

OUTPUT

```
 THE STRING TO BE ENTERED:MEDICINE
THE 2ND STRING TO BE ENTERED:MECHANICAL
THE STRINGS ARE NOT SAME
```

29. Write a C program to find the reverse of a string recursively and non recursively.

PROBLEM STATEMENT In this problem we aim to understand how to use recursions, pointers and operators by considering an example from computer language. It is required to give information such as:
string array
Once this detail is entered, The program will run to generate the reverse of the string.

ALGORITHM

START DEFINE VARIABLE: string array.
INPUT: Read the input from the keyboard.

COMPUTATION: Enter the string so that the reverse of the string is created by recursive.
DISPLAY: Reversed String.
STOP

PROGRAM

```
A) #include <stdio.h>
#include <string.h>
void reverse string(char*, int, int);

int main()
{
char stringarray[120];
printf("THE STRING TO BE ENTERED:");
scanf("%s", &stringarray);

reversestring(stringarray, 0, strlen(stringarray)-1);
printf("\nTHE STRING REVERSED IS: %s",stringarray);
return 0;
}

void reversestring(char *x, int begin, int cease)
{
char ch;
if (begin >= cease)
return;
ch = *(x+begin);
    *(x+begin) = *(x+cease);
    *(x+cease) = ch;
reversestring(x, ++begin, --cease);
}
```

OUTPUT

```
 THE STRING TO BE ENTERED:KARUMUDI
THE STRING REVERSED IS:IDMURAK
```

```
B) #include <stdio.h>
#include <string.h>
#define MAX 100
int main()
{
char string[MAX],temp;
int i=0,j=0;
printf("\nTHE STRING TO BE ENTERED");
gets(string);
j=strlen(string)-1;
```

```
while(i<j)
{
temp = string[j];
string[j]=string[i];
string[i]=temp;
i++;
j--;
}
printf("\n THE STRING REVERSED IS ");
puts(string);
}
```

OUTPUT

```
THE STRING TO BE ENTERED COMPUTER SCIENCE
 THE STRING REVERSED IS ECNEICS RETUPMOC
```

30. Create a binary tree and output the data with 3 tree traversals.

PROBLEM STATEMENT In this problem we aim to understand how to use conditions and operators by considering a concept from computer language. It is required to give information such as:
data
Once this detail is entered, The program will run to create a binary tree with 3 traversals.

ALGORITHM
START DEFINE VARIABLES: data.
INPUT: Read the input from the keyboard.
COMPUTATION: Entering the data in the nodes, So that a binary tree is generated.
DISPLAY: Traversal of the binary tree.
STOP

PROGRAM

```
 #include <stdio.h>
#include <stdlib.h>
struct node
{
int data;
struct node* left;
struct node* right;
};
struct node* newNode(int data)
{
struct node* node = (struct node*)
malloc(sizeof(struct node));
```

```
node->data = data;
node->left = NULL;
node->right = NULL;
return(node);
}
void printPostorder(struct node* node)
{
if (node == NULL)
return;
printPostorder(node->left);
printPostorder(node->right);
printf("%d ", node->data);
}
void printInorder(struct node* node)
{
if (node == NULL)
return;
printInorder(node->left);
printf("%d ", node->data);
printInorder(node->right);
}
void printPreorder(struct node* node)
{
if (node == NULL)
return;
printf("%d ", node->data);
printPreorder(node->left);
printPreorder(node->right);
}
int main()
{
struct node *root  = newNode(56);
root->left             = newNode(140);
root->right           = newNode(166);
root->left->left     = newNode(50);
root->left->right   = newNode(143);
printf("\nBINARY TREE IN PRE-ORDER TRAVERSAL IS \n");
printPreorder(root);
  printf("\nBINARY TREE IN INORDER TRAVERSAL IS \n");
printInorder(root);

printf("\nBINARY TREE IN POST-ORDER TRAVERSAL IS \n");
printPostorder(root);
```

```
getchar();
return 0;
}
```

OUTPUT

```
 BINARY TREE IN PRE-ORDER TRAVERSAL IS
 60 32 160 89 144
BINARY TREE IN INORDER TRAVERSAL IS
 160 32 89 60 144
BINARY TREE IN POST-ORDER TRAVERSAL IS
 160 89 32 144 60
```

34. Write a C program to implement the STACK operation using array as a data structure. Users must be given the following choices to perform relevant tasks.
a. Push an element on to the STACK.
b. Pop and element from the STACK.
c. Peek the STACK.
d. Display the STACK.
e. Exit the program.

PROBLEM STATEMENT In this problem we aim to understand how to use conditions, cases and operators by considering a concept from computer language. It is required to give information such as:
stack, n, x, i.
Once these details are entered, The program will run to create a stack then to perform the operations.

ALGORITHM
 START DEFINE VARIABLES: stack,n,x,i.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the elements so that the stack is generated to perform the task.
DISPLAY: Operation in the stack.
STOP

PROGRAM

```
 #include<stdio.h>
#define MAX 50
int stack[MAX],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
void peek(void);
```

```
int main()
{
top=-1;
printf("\n THE SIZE OF THE STACK TO BE ENTERED:");
scanf("%d",&n);
printf("\n\t STACK OPERATIONS APPLIED IN THIS ARRAY");
printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.PEEK\n\t 5.EXIT");
do
{
printf("\n THE CHOICE TO BE ENTERED:");
scanf("%d",&choice);
switch(choice)
{
case 1:
{
push();
break;
}
case 2:
{
pop();
break;
}
case 3:
{
display();
break;
}
case 4:
{
peek();
break;
}
case 5:
{
printf("\n\t EXIT POINT ");
break;
}
default:
{
printf ("\n\t YOU HAVE ENTERED A WRONG ELEMENT");
}
}
}
while(choice!=5);
return 0;
}
void push()
{
if(top>=n-1)
{
printf("\n\tSTACK OVER-FLOW");
}
else
{
printf(" TYPE AN ELEMENT THAT NEEDED TO BE PUSHED:");
scanf("%d",&x);
top++;
stack[top]=x;
}
}
void pop()
{
if(top<=-1)
{
printf("\n\t STACK UNDER-FLOW");
}
else
{
printf("\n\t THE ELEMENT WHICH IS POPPED %d",stack[top]);
top--;
}
}
void display()
{
if(top>=0)
{
printf("\n THE ELEMENTS IN STACK \n");
for(i=top; i>=0; i--)
printf("\n%d",stack[i]);
printf("\n ENTER NEXT CHOICE");
}
else
{
printf("\n The STACK IS EMPTY");
}
```

```
}
void peek()
{
printf("\nPEEK ELEMENT IS %d",stack[top]);
}
```

OUTPUT

```
     THE SIZE OF THE STACK TO BE ENTERED:6
     STACK OPERATIONS APPLIED IN THIS ARRAY
         1.PUSH
         2.POP
         3.DISPLAY
         4.PEEK
         5.EXIT
 THE CHOICE TO BE ENTERED:2
 TYPE AN ELEMENT THAT NEEDED TO BE PUSHED:32
  THE CHOICE TO BE ENTERED:6
   THE ELEMENTS IN STACK
    32
   ENTER NEXT CHOICE
 THE CHOICE TO BE ENTERED:2
   EXIT POINT
```

35. Write a C program to reverse a string using STACK.

PROBLEM STATEMENT In this problem we aim to understand how to use conditions and operators by considering an example from computer language. It is required to give information such as:
stack
Once this detail is entered, the program will run to create a reverse string.

ALGORITHM
START DEFINE VARIABLES: stack.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the string to generate a reversed stack.
DISPLAY: Reversed String.
STOP

PROGRAM

```
 #include <stdio.h>
#include <string.h>

#define max 100
```

```
int top,stack[max];

void push(char x)
{
if(top == max-1)
{
printf("stack overflow");
}
else
{
stack[++top]=x;
}
}
void pop()
{
printf("%c",stack[top--]);
}
main()
{
char str[60];
printf("The string to be entered : \n");
scanf("%s",&str);
int len = strlen(str);
int i;
for(i=0;i<len;i++)
push(str[i]);
for(i=0;i<len;i++)
pop();
}
```

OUTPUT

```
 The string to be entered :
honey
yenoh
```

36. Write a C program to convert the given infix expression to postfix expression using STACK.

PROBLEM STATEMENT In this problem we aim to understand how to use loops, conditions and operators by considering a concept from computer language. It is required to give information such as:
st
Once this detail is entered, The program will run to generate in-fix to post-fix expression.

ALGORITHM
 START DEFINE VARIABLES: st.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter an infix expression so that post-fix expression is generated.
DISPLAY: Post-fix expression.
STOP

PROGRAM

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#define MAX 70

char st[MAX];
int top =-1;
void push(char st[],char);
char pop(char st[]);
void InfixtoPostfix(char source[],char target[]);
int getPriority(char);
int main()
{
char infix[100], postfix[100];
   printf("\n the expression of infix to be entered");
   gets(infix);
   strcpy(postfix,"");
InfixtoPostfix(infix,postfix);
   printf("\n expression of postfix is ");
   puts(postfix);
return 0;
}
void InfixtoPostfix(char source[],char target[])
{
int i=0,j=0;
char temp;
   strcpy(target,"");
while(source[i]!='\0')
{
if(source[i]=='(')
{
push(st,source[i]);
i++;
}
else if(source[i]==')')
```

```
{
while((top!=-1)&&(st[top]!='('))
{
target[j]=pop(st);
j++;
}
if(top==-1)
{
  printf("\n incorrect expression");
  exit(1);
}
temp= pop(st);
i++;
}
else if(isdigit(source[i])|| isalpha(source[i]))
{
target[j]=source[i];
j++;
i++;
}
else if(source[i]=='+'|| source[i]=='-'|| source[i]=='*'|| source[i]== '/'|| source[i]=='%')
{
while((top!=-1)&&(st[top]!='(')&&
    (getPriority(st[top])>getPriority
    (source[i])))
{
target[j]=pop(st);
j++;
}
push(st,source[i]);
i++;
}
else
{
  printf("\n incorrect elemnet is present in the expression");
  exit(1);
}
}
while((top!=-1)&&(st[top]!='('))
{
target[j]=pop(st);
j++;
}
target[j]='\0';
}
int getPriority (char op)
{
if(op=='/'||op=='*'||op=='%')
return 1;
else if(op=='+'||op=='-')
return 0;
}
void push(char st[],char val)
{
if(top==MAX -1)
  printf("\n overflow of stack");
else
{
top++;
st[top]=val;
}
}
char pop(char st[])
{
char val=' ';
if(top==-1)
  printf("\n underflow of stack");
else
{
val=st[top];
top--;
}
return val;
}
```

OUTPUT

```
 The expression of infix to be entered (U-V)*(X-Y)
   expression of postfix is UV-XY-*
```

38. Write a C program to evaluate the given pre-fix expression and post-fix Expressions.

PROBLEM STATEMENT In this problem we aim to understand how to use cases, conditions and operators by considering an example from computer language. It is required to give information such as:
stack
Once this detail is entered, the program will run to evaluate in-fix expression.

ALGORITHM
 START DEFINE VARIABLES: stack
INPUT: Read the input from the keyboard.
COMPUTATION: Enter a pre-fix expression to generate evaluate pre-fix.
DISPLAY: Evaluate pre-fix expression.
STOP

PROGRAM
```
 #include<stdio.h>

int stack[20];
int top = -1;


void push(int x)
{
stack[++top] = x;
}

int pop()
{
return stack[top--];
}

int main()
{
char exp[20];
char *e;
int n1,n2,n3,num;
printf("ENTER THE EXPRESSION : ");
scanf("%s",exp);
e = exp;
while(*e != '\0')
{
if(isdigit(*e))
{
num = *e - 48;
push(num);
}
else
{
n1 = pop();
n2 = pop();
switch(*e)
{
case '+':
{
n3 = n1 + n2;
break;

}
case '-':
{
n3 = n2 - n1;
break;

}
case '*':
{
n3 = n1 * n2;
break;

}
case '/':
{
n3 = n2 / n1;
break;

}
}
push(n3);
}
e++;
}
printf("\nTHE RESULT OF EXPRESSION %s  =  %d\n\n",exp,pop());
return 0;
}
```

OUTPUT

```
 THE EXPRESSION TO BE ENTERED : 642*6+3
 THE RESULT OF EXPRESSION 642*6+3=3
```

37. Write a C program to convert the given in-fix expression to pre-fix expression using STACK.

PROBLEM STATEMENT In this problem we aim to understand how to use loops, conditions and operators by considering a concept from computer language. It is required to give information such as:
st
Once this detail is entered, The program will run to convert in-fix to pre-fix expression.

ALGORITHM
START DEFINE VARIABLES: st.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter an in-fix expression to generate in-fix expression.
DISPLAY: Post-fix expression.
STOP

PROGRAM
```
 #include<stdio.h>
#include<string.h>
#include<ctype.h>
#define MAX 50

char st[MAX];
int top=-1;
void reverse(char str[]);
void push(char st[],char);
char pop(char st[]);
void Infixtopostfix(char source[],char target[]);
int getPriority(char);
char infix[100],postfix[100],temp[100];
int main()
{
printf("\n ENTER THE INFIX EXPRESSION");
gets(infix);
reverse(infix);
strcpy(postfix,"");
Infixtopostfix(temp,postfix);
printf("\n THE CORRESPONDING POSTFIX EXPRESSION");
puts(postfix);
strcpy(temp,"");
reverse(postfix);
printf("\n THE PREFIX EXPRESSION");
puts(temp);
return 0;
}
void reverse(char str[])
```

```
{
int len,i=0,j=0;;
len=strlen(str);
j=len-1;
while(j>=0)
{
if(str[j]=='(')
  temp[i]=')';
else if(str[j]==')')
  temp[i]='(';
else
temp[i]=str[j];
i++;
j--;
}
temp[i]='\0';
}
void Infixtopostfix(char source[], char target[])
{
int i=0,j=0;
char temp;
strcpy(target,"");
while(source[i]!='\0')
{
if(source[i]=='(')
{
push(st, source[i]);
i++;
}
else if(source[i]==')')
{
while((top!=-1)&&(st[top]!='('))
{
target[j]=pop(st);
j++;
}
if(top==-1)
{
printf("\n INCORRECT EXPRESSION");
exit(1);
}
temp=pop(st);
i++;
```

```
}
else if(isdigit(source[i])||isalpha(source[i]))
{
target[j]= source[i];
j++;
i++;
}
else if(source[i]=='+'||source[i]=='-'
        ||source[i]=='*'||source[i]
        =='/'||source[i]=='%')
{
while((top!=-1)&&(st[top]!='(')
     &&(getPriority(st[top])>
     getPriority(source[i])))
{
target[j]= pop(st);
j++;
}
push(st, source[i]);
i++;
}
else
{
printf("\n INCORRECT ELEMENTS IN EXPRESSION");
exit(1);
}
}
while((top!=-1)&&(st[top]!='('))
{
target[j]= pop(st);
j++;
}
target[j]='\0';
}
int getPriority(char op)
{
if(op=='/'||op=='*'||op=='%')
return 1;
else if(op=='+'||op=='-')
return 0;
}
void push(char st[], char val)
{
```

```
if(top==MAX -1)
printf("\n STACK OVER-FLOW");
else
{
top++;
st[top]=val;
}
}
char pop(char st[])
{
char val= ' ';
if(top==-1)
printf("\n STACK UNDER-FLOW");
else
{
val=st[top];
top--;
}
return val;
}
```

OUTPUT

```
 ENTER THE INFIX EXPRESSION  U+V-X*Y
THE CORRESPONDING POSTFIX EXPRESSION   X*VU+-
THE PREFIX EXPRESSION -+UV*XY
```

39. Write a C program to implement a Linear-Queue, user must choose the following options:

a. Add an element to the Queue – EnQueue.

b. Remove an element from the Queue – DeQueue.

c. Display the elements of the Queue.

d. Terminate the program. #include<stdio.h>

PROBLEM STATEMENT In this problem we aim to understand how to use loops, cases, conditions and operators by considering a concept from computer language. It is required to give information such as:
queue
Once this detail is entered, The program will run to generate a linear-queue to perform operations.

ALGORITHM

START DEFINE VARIABLES: queue.

INPUT: Read the input from the keyboard.

COMPUTATION: Enter the elements so that a linear queue is created.

DISPLAY: Operations in linear queue.
STOP

PROGRAM

```
#define MAX 50

int queue[MAX];
int front=-1,rear=-1;
void insert(void);
int delete_element(void);
int peep(void);
void display(void);
int main()
{
int option,val;
do
{
printf("\n\n*****MAIN MENU*****");
printf("\n 1. ELEMENT TO BE ADDED");
printf("\n 2. ELEMENT TO BE DELETED");
printf("\n 3. PEEK");
printf("\n 4. QUEUE TO BE DISPLAYED");
printf("\n 5. EXIT");
printf("\n ********************");
printf("\n\n ENTER THE OPTION");
scanf("%d",&option);
switch(option)
{
case 1:
insert();
break;

case 2:
val=delete_element();
if(val!=-1)
printf("\n THE NUMBER TO BE DELETED IS %d",val);
break;

case 3:
val= peep();
if(val!=-1)
printf("\n THE FIRST ELEMENT IN THE QUEUE IS %d",val);
break;
case 4:
display();
break;
}
}while(option!=5);
return 0;
}
void insert()
{
int num;
printf("\n ENTER THE ELEMENT TO BE ADDED ");
scanf("%d",&num);
if(rear==MAX-1)
printf("\n OVER-FLOW");
else if(front==-1&&rear==-1)
front=rear=0;
else
rear++;
queue[rear]=num;
}
int delete_element()
{
int val;
if(front==-1 || front>rear)
{
printf("\n UNDER-FLOW");
return -1;
}
else
{
val=queue[front];
front++;
if(front>rear)
front=rear=-1;
return val;
}
}
int peep()
{
if(front==-1 || front> rear)
{
printf("\n QUEUE IS EMPTY");
```

```
return -1;
}
else
{
return queue[front];
}
}
void display()
{
int i;
printf("\n");
if(front==-1 || front > rear)
printf("\n QUEUE IS EMPTY");
else
{
for(i=front;i<=rear;i++)
printf("\t %d",queue[i]);
}
}
```

OUTPUT

```
 *****MAIN MENU*****
 1. ELEMENT TO BE ADDED
 2. ELEMENT TO BE DELETED
 3. PEEK
 4. QUEUE TO BE DISPLAYED
 5. EXIT
 ********************
 ENTER THE OPTION1
 ENTER THE ELEMENT TO BE ADDED 66
 *****MAIN MENU*****
 1. ELEMENT TO BE ADDED
 2. ELEMENT TO BE DELETED
 3. PEEK
 4. QUEUE TO BE DISPLAYED
 5. EXIT
 ********************
  ENTER THE OPTION3
THE FIRST ELEMENT IN THE QUEUE IS 66
 *****MAIN MENU*****
 1. ELEMENT TO BE ADDED
 2. ELEMENT TO BE DELETED
 3. PEEK
 4. QUEUE TO BE DISPLAYED
 5. EXIT
 ********************
  ENTER THE OPTION4
                66
 *****MAIN MENU*****
 1. ELEMENT TO BE ADDED
 2. ELEMENT TO BE DELETED
 3. PEEK
 4. QUEUE TO BE DISPLAYED
 5. EXIT
 ********************
 ENTER THE OPTION5
```

40. Write a C program to implement a Circular-Queue, user must choose the following options:

a. Add an element to the Queue – EnQueue.
b. Remove an element from the Queue – DeQueue.
c. Display the elements of the Queue.
d. Terminate the program.

PROBLEM STATEMENT In this problem we aim to understand how to use loops, cases, conditions and operators by considering a concept from computer language. It is required to give information such as:
queue
Once this detail is entered, The program will run to create a Circular-queue to perform operations.

ALGORITHM
START DEFINE VARIABLES: queue.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the elements so that a linear queue is generated.
DISPLAY: Operations in linear queue.
STOP

PROGRAM

```c
 #include<stdio.h>
#define MAX 50

void insertq(int[], int);
void deleteq(int[]);
void display(int[]);

int front =  - 1;
int rear =  - 1;
int main()
{
int n, ch;
int queue[MAX];
do
{
printf("\n\n CIRCULAR QUEUE CHOICES:\n1. ENQUEUE \n2. DEQUEUE\n3. DISPLAY\n0.
EXIT");
printf("\nPRESS THE CHOICE: ");
scanf("%d", &ch);
switch (ch)
{
case 1:
printf("\n THE NUMBER TO BE ENTERED: ");
scanf("%d", &n);
insertq(queue, n);
break;

case 2:
deleteq(queue);
break;

case 3:
display(queue);
break;



}
}while (ch != 0);
}

void insertq(int queue[], int item)
{
if ((front == 0 && rear == MAX - 1) || (front == rear + 1))
{
printf("QUEUE OVER-FLOW");
return;
}
else if (rear ==  - 1)
{
rear++;
front++;
```

```
}
else if (rear == MAX - 1 && front > 0)
{
rear = 0;
}
else
{
rear++;
}
queue[rear] = item;
}

void display(int queue[])
{
int i;
printf("\n");
if (front > rear)
{
for (i = front; i < MAX; i++)
{
printf("%d ", queue[i]);
}
for (i = 0; i <= rear; i++)
printf("%d ", queue[i]);
}
else
{
for (i = front; i <= rear; i++)
printf("%d ", queue[i]);
}
}

void deleteq(int queue[])
{
if (front ==  - 1)
{
printf("QUEUE UNDER-FLOW ");
}
else if (front == rear)
{
printf("\n %d REMOVED", queue[front]);
front =  - 1;
rear =  - 1;
```

```
}
else
{
printf("\n %d REMOVED", queue[front]);
front++;
}
}
```

OUTPUT

```
 CIRCULAR QUEUE CHOICES:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
0. EXIT
PRESS THE CHOICE: 1
THE NUMBER TO BE ENTERED: 54
 CIRCULAR QUEUE CHOICES:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
0. EXIT
PRESS THE CHOICE:3
54
 CIRCULAR QUEUE CHOICES:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
0. EXIT
PRESS THE CHOICE: 3
54
CIRCULAR QUEUE CHOICES:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
0. EXIT
PRESS THE CHOICE : 2
  54 REMOVED
 CIRCULAR QUEUE CHOICES:
 1. ENQUEUE
  2. DEQUEUE
 3. DISPLAY
0. EXIT
 PRESS THE CHOICE: 0
```

41. Write a C program to create a single linked list with 5 nodes. (5 integers are taken from user input) and display the linked-list elements.

PROBLEM STATEMENT In this problem we aim to understand how to use conditions and operators by considering a concept from computer language. It is required to give information such as:
num, n, total nodes
Once these details are entered, The program will run to generate a single linked list.

ALGORITHM
START DEFINE VARIABLES: num, n, total nodes.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the data of the nodes, So that a single linked list is generated.
DISPLAY: Single linked list with 5 nodes.
STOP

PROGRAM
```
 #include <stdio.h>
#include <stdlib.h>
struct node
{
int num;
struct node *nextptr;
}*stnode;
void createNodeList(int n);
int NodeCount();
void displayList();
int main()
{
int n,totalNode;
printf("\n\n CREATE A SINGLE LINKED LIST THEN COUNT THE TOTAL NODES:\n");
printf("-----------------------------------------------------------\n");
printf(" ENTER THE VALUE OF NODES: ");
scanf("%d", &n);
createNodeList(n);
printf("\n VALUES GIVEN IN THE SLL ARE : \n");
displayList();
totalNode = NodeCount();
printf("\n TOTAL NUMBER OF NODES = %d\n", totalNode);
return 0;
}

void createNodeList(int n)
```

```
{
struct node *fnNode, *tmp;
int num, i;
stnode = (struct node *)malloc(sizeof(struct node));
if(stnode == NULL)
{
printf(" MEMORY CAN'T BE GIVEN.");
}
else
{
printf(" GIVEN DATA FOR NODE 1 : ");
scanf("%d", &num);
stnode-> num = num;
stnode-> nextptr = NULL;
tmp = stnode;
for(i=2; i<=n; i++)
{
fnNode = (struct node *)malloc(sizeof(struct node));
if(fnNode == NULL)
{
printf(" MEMORY CAN'T BE GIVEN.");
break;
}
else
{
printf(" GIVEN DATA FOR NODE %d : ", i);
scanf(" %d", &num);
fnNode->num = num;
fnNode->nextptr = NULL;
tmp->nextptr = fnNode;
tmp = tmp->nextptr;
}
}
}
}

int NodeCount()
{
int ctr = 0;
struct node *tmp;
tmp = stnode;
while(tmp != NULL)
{
```

```
ctr++;
tmp = tmp->nextptr;
}
return ctr;
}

void displayList()
{
struct node *tmp;
if(stnode == NULL)
{
printf(" NO DATA FOUND IN THE LIST.");
}
else
{
tmp = stnode;
while(tmp != NULL)
{
printf(" DATA = %d\n", tmp->num);
tmp = tmp->nextptr;
}
}
}
```

OUTPUT

```
CREATE A SINGLE LINKED LIST THEN COUNT THE TOTAL NODES:
-------------------------------------------------------
 ENTER THE VALUE OF NODES: 3
 GIVEN DATA FOR NODE 1 : 10
 GIVEN DATA FOR NODE 2 : 20
 GIVEN DATA FOR NODE 3 : 30

  VALUES GIVEN IN THE SLL ARE :
 DATA = 10
 DATA = 20
 DATA = 30
  TOTAL NUMBER OF NODES = 3
```

42. Write a C program to search an element in a singly-linked

PROBLEM STATEMENT In this problem we aim to understand how to use conditions and operators by considering a concept from computer language. It is required to give information such as:
num, n, FindElem

Once these details are entered, The program will create a single linked list and search an element in the list.

ALGORITHM
 START DEFINE VARIABLES: num, n, FindElem.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the elements in in the singly linked list, to search the element in the list.
DISPLAY: Searched element.
STOP

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
int num;
struct node *nextptr;
}

stnode, *ennode;

int FindElement(int);
void main()
{
int n,i,FindElem,FindPlc;
stnode.nextptr=NULL;
ennode=&stnode;
printf("\n\n LINKED LIST : SEARCH AN ELEMENT IN SLL :\n");
printf("--------------------------------------------\n");

printf(" ENTER THE VALUE OF NODES : ");
scanf("%d", &n);
printf("\n");
for(i=0;i< n;i++)
{
ennode->nextptr=(struct node *)malloc(sizeof(struct node));
printf(" INPUT DATA FOR NODE %d : ",i+1);
scanf("%d",&ennode->num);
ennode=ennode->nextptr;
}
ennode->nextptr=NULL;
printf("\n ENTERED ELEMENTS IN THE SLL ARE :\n");
ennode=&stnode;
while(ennode->nextptr!=NULL)
{
printf(" DATA = %d\n",ennode->num);
ennode=ennode->nextptr;
}

printf("\n");
printf(" TYPE THE ELEMENT TO BE SEARCHED : ");
scanf("%d",&FindElem);
FindPlc=FindElement(FindElem);
if(FindPlc<=n)
printf(" ELEMENT FOUND AT NODE %d \n\n",FindPlc);
else
printf(" ELEMENT DOES'NT EXISTS IN LINKED LIST.\n\n");
}

int FindElement(int FindElem)
{
int ctr=1;
ennode=&stnode;
while(ennode->nextptr!=NULL)
{
if(ennode->num==FindElem)
break;
else
ctr++;
ennode=ennode->nextptr;
}
return ctr;
}list.
```
OUTPUT
```
  LINKED LIST : SEARCH AN ELEMENT IN SLL :
--------------------------------------------
 ENTER THE VALUE OF NODES : 2

 INPUT DATA FOR NODE 1 : 45
 INPUT DATA FOR NODE 2 : 789
 ENTERED ELEMENTS IN THE SLL ARE :
 DATA = 45
 DATA = 789
  TYPE THE ELEMENT TO BE SEARCHED : 45
 ELEMENT FOUND AT NODE 1
```

44. Write a C program to create a doubly linked list with 5 nodes.

PROBLEM STATEMENT In this problem we aim to understand how to use conditions and operators by considering a concept from computer language. It is required to give information such as:
num, n
Once these details are entered, The program will run to create doubly linked list.

ALGORITHM
START DEFINE VARIABLES: num, n.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the data of the nodes, to generate the doubly linked list
DISPLAY: Doubly linked list with 5 nodes.
STOP

PROGRAM
```
 #include <stdio.h>
#include <stdlib.h>

struct node {
int num;
struct node * preptr;
struct node * nextptr;
}*stnode, *ennode;


void DlListcreation(int n);
void displayDlList();

int main()
{
int n;
stnode = NULL;
ennode = NULL;
printf("\n\n DOUBLY LINKED LIST : GENERATE AND DISPLAY A DLL :\n");
printf("-------------------------------------------------------\n");

printf(" ENTER THE NUMBER OF NODES : ");
scanf("%d", &n);

DlListcreation(n);
```

```
displayDlList();
return 0;
}

void DlListcreation(int n)
{
int i, num;
struct node *fnNode;

if(n >= 1)
{
stnode = (struct node *)malloc(sizeof(struct node));

if(stnode != NULL)
{
printf(" DATA ENTERED FOR NODE 1 : ");
scanf("%d", &num);

stnode->num = num;
stnode->preptr = NULL;
stnode->nextptr = NULL;
ennode = stnode;

for(i=2; i<=n; i++)
{
fnNode = (struct node *)malloc(sizeof(struct node));
if(fnNode != NULL)
{
printf(" DATA ENTERED FOR NODE %d : ", i);
scanf("%d", &num);
fnNode->num = num;
fnNode->preptr = ennode;
fnNode->nextptr = NULL;

ennode->nextptr = fnNode;
ennode = fnNode;
}
else
{
printf(" MEMORY CAN'T BE GIVEN.");
break;
}
}
}
else
{
printf(" MEMORY CAN'T BE GIVEN.");
}
}
}
void displayDlList()
{
struct node * tmp;
int n = 1;
if(stnode == NULL)
{
printf(" NO DATA FOUND IN THE LIST.");
}
else
{
tmp = stnode;
printf("\n\n DATA ENTERED ON THE LIST:\n");

while(tmp != NULL)
{
printf(" NODE %d : %d\n", n, tmp->num);
n++;
tmp = tmp->nextptr;
}
}
}
```

OUTPUT

```
 DOUBLY LINKED LIST : GENERATE AND DISPLAY A DLL :
-----------------------------------------------------
 ENTER THE NUMBER OF NODES : 4
 DATA ENTERED FOR NODE 1 : 10
 DATA ENTERED FOR NODE 2 : 20
 DATA ENTERED FOR NODE 3 : 30
 DATA ENTERED FOR NODE 4 : 40
  DATA ENTERED ON THE LIST:
 NODE 1 : 10
 NODE 2 : 20
 NODE 3 : 30
 NODE 4 : 40
```

45. Write a C program to create a circular linked list with 5 nodes.

PROBLEM STATEMENT In this problem we aim to understand how to use conditions and operators by considering a concept from computer language. It is required to give information such as:
num, n
Once these details are entered, The program will create a circular linked list.

ALGORITHM
START DEFINE VARIABLES: num, n.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the data of the nodes, to generate a circular linked list.
DISPLAY: Circular linked list with 5 nodes.
STOP

PROGRAM
```
#include <stdio.h>
#include <stdlib.h>

struct node {
int num;
struct node * nextptr;
}*stnode;


void ClListcreation(int n);
void displayClList();

int main()
{
int n;
stnode = NULL;
printf("\n\n CIRCULAR LINKED LIST : GENERATE AND DISPLAY A CLL:\n");
printf("-----------------------------------------------------\n");

printf(" TYPE THE NUMBER OF NODES : ");
scanf("%d", &n);

ClListcreation(n);
displayClList();
return 0;
}

void ClListcreation(int n)
```

```
{
int i, num;
struct node *preptr, *newnode;

if(n >= 1)
{
stnode = (struct node *)malloc(sizeof(struct node));

printf(" ENTERED DATA FOR NODE 1 : ");
scanf("%d", &num);
stnode->num = num;
stnode->nextptr = NULL;
preptr = stnode;
for(i=2; i<=n; i++)
{
newnode = (struct node *)malloc(sizeof(struct node));
printf(" ENTERED DATA FOR NODE %d : ", i);
scanf("%d", &num);
newnode->num = num;
newnode->nextptr = NULL;
preptr->nextptr = newnode;
preptr = newnode;
}
preptr->nextptr = stnode;
}
}

void displayClList()
{
struct node *tmp;
int n = 1;

if(stnode == NULL)
{
printf(" NONE DATA WAS FOUND IN THE LIST.");
}
else
{
tmp = stnode;
printf("\n\n DATA ALLOCATED IN THE LIST:\n");

do {
printf(" DATA %d = %d\n", n, tmp->num);
tmp = tmp->nextptr;
n++;
}while(tmp != stnode);
}
}
```

OUTPUT

```
 CIRCULAR LINKED LIST : GENERATE AND DISPLAY A CLL:
-------------------------------------------------------
 TYPE THE NUMBER OF NODES : 3
 ENTERED DATA FOR NODE 1 :20
ENTERED DATA FOR NODE 2 : 40
 ENTERED DATA FOR NODE 3 : 60
 DATA ALLOCATED IN THE LIST:
 DATA 1 = 20
 DATA 2 = 40
 DATA 3 = 60
```

46. Write a C program to implement the stack using linked list.

PROBLEM STATEMENT In this problem we aim to understand how to use loops, cases, conditions and operators by considering a concept from computer language. It is required to give information such as:
no, ch, e
Once these details are entered, The program will run to generate a stack using a linked list.

ALGORITHM
START DEFINE VARIABLES: no, ch, e.
INPUT: Read the input from the keyboard.
COMPUTATION: Enter the elements to generate a stack using a linked list.
DISPLAY: Stack
STOP

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
int info;
struct node *ptr;
}*top,*top1,*temp;

int topelement();
```

```
void push(int data);
void pop();
void empty();
void display();
void destroy();
void stack_count();
void create();

int count = 0;

void main()
{
int no, ch, e;

printf("\n 1 - PUSH THE ELEMENT");
printf("\n 2 - POP THE ELEMENT");
printf("\n 3 - SHOW THE TOP ELEMENT");
printf("\n 4 - EMPTY THE LIST");
printf("\n 5 - EXIT");
printf("\n 6 - DISPLAY THE LIST");
printf("\n 7 - COUNT THE ELEMENT IN STACK");
printf("\n 8 - DESTROY THE STACK");

create();

while (1)
{
printf("\n TYPE YOUR CHOICE: ");
scanf("%d", &ch);

switch (ch)
{
case 1:
printf("ENTER THE ELEMENT: ");
scanf("%d", &no);
push(no);
break;

case 2:
pop();
break;

case 3:
```

```
if (top == NULL)
printf("THERE IS NO ELEMENT IN THE STACK");
else
{
e = topelement();
printf("\n ELEMENT IN THE TOP: %d", e);
}
break;

case 4:
empty();
break;

case 5:
exit(0);

case 6:
display();
break;

case 7:
stack_count();
break;

case 8:
destroy();
break;

default :
printf(" WRONG CHOICE,ENTER THE CORRECT CHOICE");
break;
}
}
}



void create()
{
top = NULL;
}
```

```
void stack_count()
{
printf("\n TOTAL ELEMENTS IN STACK: %d", count);
}


void push(int data)
{
if (top == NULL)
{
top =(struct node *)malloc(1*sizeof(struct node));
     top->ptr = NULL;
     top->info = data;
}
else
{
temp =(struct node *)malloc(1*sizeof(struct node));
    temp->ptr = top;
    temp->info = data;
    top = temp;
}
count++;
}

void display()
{
top1 = top;

if (top1 == NULL)
{
printf("STACK IS EMPTY");
return;
}

while (top1 != NULL)
{
printf("%d ", top1->info);
    top1 = top1->ptr;
}
}


void pop()
```

```
{
top1 = top;

if (top1 == NULL)
{
printf("\n ERROR");
return;
}
else
top1 = top1->ptr;
printf("\n POPPED ELEMENT : %d", top->info);
free(top);
top = top1;
count--;
}



int topelement()
{
   return(top->info);
}


void empty()
{
if (top == NULL)
printf("\n STACK IS EMPTY");
else
printf("\n STACK IS NOT EMPTY WITH %d ELEMENTS", count);
}


void destroy()
{
top1 = top;

while (top1 != NULL)
{
top1 = top->ptr;
free(top);
top = top1;
top1 = top1->ptr;
}
free(top1);
top = NULL;
printf("\n  STACK ELEMENTS ARE DESTROYED");
count = 0;
}
```

OUTPUT

```
 1 - PUSH THE ELEMENT
 2 - POP THE ELEMENT
 3 - SHOW THE TOP ELEMENT
 4 - EMPTY THE LIST
 5 - EXIT
 6 - DISPLAY THE LIST
 7 - COUNT THE ELEMENT IN STACK
 8 - DESTROY THE STACK
 TYPE YOUR CHOICE: 1
ENTER THE ELEMENT: 43

 TYPE YOUR CHOICE: 1
ENTER THE ELEMENT: 52

 TYPE YOUR CHOICE: 4

 STACK IS NOT EMPTY WITH 2 ELEMENTS
 TYPE YOUR CHOICE: 1
ENTER THE ELEMENT: 65

 TYPE YOUR CHOICE: 8

  STACK ELEMENTS ARE DESTROYED
 TYPE YOUR CHOICE: 5
```

47. Write a C program to implement the queue using a linked list.

PROBLEM STATEMENT In this problem we aim to understand how to use loops, cases, conditions and operators by considering a concept from computer language. It is required to give information such as:
no, ch, e
Once these details are entered, The program will run to generate a queue using a linked list.

ALGORITHM
 START DEFINE VARIABLES: no, ch, e.
INPUT: Read the input from the keyboard.

COMPUTATION: Enter the elements to create a queue using a linked list.
DISPLAY: Queue
STOP

PROGRAM
```
 #include <stdio.h>
#include <stdlib.h>

struct node
{
int info;
struct node *ptr;
}*front,*rear,*temp,*front1;

int frontelement();
void enq(int data);
void deq();
void empty();
void display();
void create();
void queuesize();

int count = 0;

void main()
{
int no, ch, e;
printf("\n------SELECT--------");
printf("\n 1 - ENQUEUE");
printf("\n 2 - DEQUEUE");
printf("\n 3 - FIRST ELEMENT");
printf("\n 4 - EMPTY");
printf("\n 5 - EXIT");
printf("\n 6 - DISPLAY THE LIST");
printf("\n 7 - SIZE OF THE QUEUE");
create();
while (1)
{
printf("\n ENTER THE CHOICE : ");
scanf("%d", &ch);
switch (ch)
{
```

```
case 1:
printf("TYPE THE DATA : ");
scanf("%d", &no);
enq(no);
break;

case 2:
deq();
break;

case 3:
e = frontelement();
if (e != 0)
printf("FIRST ELEMENT : %d", e);
else
printf("\n ABSENCE OF FIRST ELEMENT IN QUEUE");
break;

case 4:
empty();
break;

case 5:
exit(0);

case 6:
display();
break;

case 7:
queuesize();
break;

default:
printf("WRONG CHOICE,ENTER THE CORRECT CHOICE");
break;
}
}
}


void create()
{
```

```
front = rear = NULL;
}

void queuesize()
{
printf("\n SIZE OF QUEUE : %d", count);
}

void enq(int data)
{
if (rear == NULL)
{
rear = (struct node *)malloc(1*sizeof(struct node));
        rear->ptr = NULL;
        rear->info = data;
        front = rear;
}
else
{
temp=(struct node *)malloc(1*sizeof(struct node));
      rear->ptr = temp;
      temp->info = data;
      temp->ptr = NULL;

        rear = temp;
}
count++;
}

void display()
{
front1 = front;
if ((front1 == NULL) && (rear == NULL))
{
printf("QUEUE IS EMPTY");
return;
}
while (front1 != rear)
{
printf("%d ", front1->info);
        front1 = front1->ptr;
}
if (front1 == rear)
```

```
printf("%d", front1->info);
}

void deq()
{
front1 = front;

if (front1 == NULL)
{
printf("\n ERROR");
return;
}
else
if (front1->ptr != NULL)
{
front1 = front1->ptr;
printf("\n DEQUED ELEMENT : %d", front->info);
free(front);
front = front1;
}
else
{
printf("\n DEQUED ELEMENT : %d", front->info);
free(front);
front = NULL;
rear = NULL;
}
count--;
}

int frontelement()
{
if ((front != NULL) && (rear != NULL))
return(front->info);
else
return 0;
}

void empty()
{
if ((front == NULL) && (rear == NULL))
printf("\n QUEUE IS EMPTY");
else
```

```
printf("QUEUE IS NOT EMPTY");
}
```

OUTPUT

```
 ------SELECT--------
 1 - ENQUEUE
 2 - DEQUEUE
 3 - FIRST ELEMENT
 4 - EMPTY
 5 - EXIT
 6 - DISPLAY THE LIST
 7 - SIZE OF THE QUEUE
 ENTER THE CHOICE : 1
TYPE THE DATA : 75
 ENTER THE CHOICE : 1
TYPE THE DATA : 54
 ENTER THE CHOICE : 3
FIRST ELEMENT : 75
 ENTER THE CHOICE : 5
```