1. **Design, Develop and Implement a menu driven Program in C for the following Array Operations**
   a. Creating an Array of N Integer Elements
   b. Inserting an Element (ELEM) at a given valid Position (POS)
   c. Deleting an Element at a given valid Position POS) c. Display of Array Elements
   d. Display of Array Elements with Suitable Headings
   e. Exit.
   Support the program with functions for each of the above operations.

```
#include<stdio.h>
int a[20];
int n, val, i, pos, choice;
void create();
void display();
void insert();
void delet();
int main()
{
while(1)
{
printf("\n\n--------MENU-----------\n");
printf("1.CREATE\n");
printf("2.DISPLAY\n");
printf("3.INSERT\n");
printf("4.DELETE\n");
printf("5.EXIT\n");
printf("-----------------------");
printf("\nENTER YOUR CHOICE:\t");
scanf("%d",&choice);
switch(choice)
{
case 1: create();
break;
case 2:display();
break;
case 3:insert();
break;
case 4:if(n==0)
{
printf("Array is empty\n"); break;
}
else
delet();
break;
case 5:exit(0);
break;
default:printf("\nInvalid choice:\n");
break;
}
}
return 0;
}
void create()
{
printf("\nEnter the size of the array elements:\t");
scanf("%d",&n);
printf("\nEnter the elements for the array:\n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
}
void display()
{
int i;
printf("\nThe array elements are:\n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
}
void insert()
{
printf("\nEnter the position for the new element:\t");
scanf("%d",&pos);
printf("\nEnter the element to be inserted :\t");
scanf("%d",&val);
for(i=n-1;i>=pos-1;i--)
{
a[i+1]=a[i];
}
a[pos-1]=val;
n=n+1;
}
void delet()
{
printf("\nEnter the position of the element to be deleted:\t");
scanf("%d",&pos);
val=a[pos-1];
for(i=pos-1;i<n-1;i++)
{
a[i]=a[i+1];
}
n=n-1;
printf("\nThe deleted element is =%d",val);
}
```

OUTPUT:

```
        cc 1.c OR gcc 1.c
        ./a.out

--------MENU-----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
-----------------------
ENTER YOUR CHOICE: 2
The array elements are:
--------MENU-----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
-----------------------
ENTER YOUR CHOICE: 4
Array is empty
--------MENU-----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
-----------------------
ENTER YOUR CHOICE: 1
Enter the size of the array elements: 3
Enter the elements for the array:
1
9
66
--------MENU-----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
-----------------------
ENTER YOUR CHOICE: 2
The array elements are:
1 9 66
--------MENU-----------
1.CREATE
2.DISPLAY
```

```
3.INSERT
4.DELETE
5.EXIT
-----------------------
ENTER YOUR CHOICE: 3
Enter the position for the new element: 1
Enter the element to be inserted : 100
--------MENU-----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
-----------------------
ENTER YOUR CHOICE: 2
The array elements are:
100 1 9 66
--------MENU-----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
-----------------------
ENTER YOUR CHOICE: 4
Enter the position of the element to be deleted: 3
The deleted element is =9
--------MENU-----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
-----------------------
ENTER YOUR CHOICE: 2
The array elements are:
100 1 66
--------MENU-----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
-----------------------
ENTER YOUR CHOICE: 5
```

2. **Design, Develop and Implement a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)**
   a. Push an Element on to Stack
   b. Pop an Element from Stack
   c. Demonstrate Overflow and Underflow situations on Stack
   d. Display the status of Stack
   e. Exit
   Support the program with appropriate functions for each of the above operations

```
#include<stdio.h>
#include<string.h>
#define max_size 5
int stack[max_size],top=-1,flag=1;
int i,temp,item,rev[max_size],num[max_size];
void push();
void pop();
void display();
void pali();
void main()
{
int choice;
printf("\n\n-------- STACK OPERATIONS----------- \n");
printf("1.Push\n");
printf("2.Pop\n");
printf("3.Palindrome\n");
printf("4.Display\n");
printf("5.Exit\n");
printf("----------------------- ");
while(1)
{
printf("\nEnter your choice:\t");
scanf("%d",&choice);
switch(choice)
{
case 1: push();
break;
case 2: pop();
if(flag)
printf("\nThe poped element: %d\t",item);
temp=top;
break;
case 3: display();
break;
case 4: exit(0);
break;
default: printf("\nInvalid choice:\n");
break;
}
}
}
void push()
{
if(top==(max_size-1))
{
printf("\nStack Overflow:");
}
else
{
printf("Enter the element to be inserted:\t");
scanf("%d",&item);
top=top+1;
stack[top]=item;
}
temp=top;
}
void pop()
{
if(top==-1)
{
printf("Stack Underflow:");
flag=0;
}
else
{
item=stack[top];
top=top-1;
}
}
void display()
{
int i;
top=temp;
if(top==-1)
{
printf("\nStack is Empty:");
}
else
{
printf("\nThe stack elements are:\n" );
for(i=top;i>=0;i--)
{
printf("%d\n",stack[i]);
}
}
}
```

OUTPUT:

```
        cc 1.c OR gcc 1.c
        ./a.out
```

```
- STACK OPERATIONS-----------
1.Push
2.Pop
3.Display
4.Exit
-----------------------
Enter your choice: 2
Stack Underflow:
Enter your choice: 3
Stack is Empty:
Enter your choice: 1
Enter the element to be inserted: 25
Enter your choice: 1
Enter the element to be inserted: 45
Enter your choice: 3
The stack elements are:
45
25
Enter your choice: 2
Enter your choice: 3
The stack elements are:
25
Enter your choice: 2
Enter your choice: 2
Stack Underflow:
Enter your choice: 1
Enter the element to be inserted: 1
Enter your choice: 1
Enter the element to be inserted: 1
Enter your choice: 3
The stack elements are:
1
1
Enter your choice: 2
Enter your choice: 1
Enter the element to be inserted: 2
Enter your choice: 1
Enter the element to be inserted: 1
Enter your choice: 2
Enter your choice: 3
The stack elements are:
2
1
```

3. Design, Develop and Implement a Program in C for the following Stack Applications
a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^
b. Solving Tower of Hanoi problem with n disks

a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^

```
#include<stdio.h>
#include<math.h>
#include<string.h>
#includde<ctype.h>
float compute(char symbol, float op1,float op2)
{
switch(symbol)
{
case '+': return op1+op2;
case '-': return op1-op2;
case '*': return op1*op2;
case '/': return op1/op2;
case '$':
case '^': return pow(op1,op2);
default: return 0;
}
}
void main()
{
float s[20],res,op1,op2;
int top,i;
char postfix[20],symbol;
printf("\n enter the postfix expression:\n");
scanf("%s", postfix);
top=-1;
for(i=0;i<strlen(postfix);i++)
{
symbol=postfix[i];
if(isdigit(symbol))
{
s[++top]=symbol-'0';
}
else
{
op2=s[top--];
op1=s[top--];
res=compute(symbol,op1,op2);
s[++top]=res;
}
}
res=s[top--];
printf("\n The result is:%f\n",res);
}
```

OUTPUT:

```
cc 1.c OR gcc 1.c
./a.out
```

```
Enter the postfix expression:
67+
The result is:13.000000

Enter the postfix expression:
52-
The result is:3.000000

Enter the postfix expression:
78*
The result is:56.000000

Enter the postfix expression:
98*
The result is:72.000000

Enter the postfix expression:
755/+4-
The result is:4.000000

Enter the postfix expression:
123*+55/-6-
The result is:0.000000

Enter the postfix expression:
925*+55/-
The result is:18.000000
```

b. Solving Tower of Hanoi problem with n disks

```
#include<stdio.h>
int count=0,n;
int tower(int n,char s, char t, char d)
{
if(n==1)
{
printf("\n move disk 1 from %c to %c\n",s,d);
count++;
return 0;
}
tower(n-1,s,d,t);
printf("\n move %d from %c to %c\n",n,s,d);
count++;
tower(n-1,t,s,d);
return 0;
}
void main()
{
printf("\n enter the number of discs:\n");
scanf("%d",&n);
printf("\n...\n");
tower(n,'A','B','C');
printf("\n...\n");
printf("\n total of %d disk takes %d moves\n\n",n,count);
}
```

OUTPUT:

```
        cc 2.c OR gcc 2.c
        ./a.out
```

```
enter the number of discs: 2
...
move disk 1 from A to B
move 2 from A to C
move disk 1 from B to C
...
total of 2 disk takes 3 moves
CRC@CRC-MS-7A15:~/CRC$ ./a.out
enter the number of discs: 3
...
move disk 1 from A to C
move 2 from A to B
move disk 1 from C to B
move 3 from A to C
move disk 1 from B to A
move 2 from B to C
move disk 1 from A to C
...
total of 3 disk takes 7 moves
```

4. Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers
a. Create a BST of N Integers
b. Traverse the BST in Inorder, Preorder and Post Order

```
#include <stdio.h>
#include <stdlib.h>
struct BST
{
int data;
struct BST *left;
struct BST *right;
};
typedef struct BST NODE;
NODE *node;
NODE* createtree(NODE *node, int data)
{
if (node == NULL)
{
NODE *temp;
temp= (NODE*)malloc(sizeof(NODE));
temp->data = data;
temp->left = temp->right = NULL;
return temp;
}
if (data < (node->data))
{
node->left = createtree(node->left, data);
}
else if (data > node->data)
{
node -> right = createtree(node->right, data);
}
return node;
}
void inorder(NODE *node)
{
if(node != NULL)
{
inorder(node->left);
printf("%d\t", node->data);
inorder(node->right);
}
}
void preorder(NODE *node)
{
if(node != NULL)
{
```

```
printf("%d\t", node->data);
preorder(node->left);
preorder(node->right);
}
}
void postorder(NODE *node)
{
if(node != NULL)
{
postorder(node->left);
postorder(node->right);
printf("%d\t", node->data);
}
}
void main()
{
int data, ch, i, n;
NODE *root=NULL;
while (1)
{
printf("\n1.Insertion in Binary Search Tree");
printf("\n2.Inorder\n3.Preorder\n4.Postorder\n5.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch (ch)
{
case 1: printf("\nEnter N value: " );
scanf("%d", &n);
printf("\nEnter-the-values-to-create-BSTn like(6,9,5,2,8,15,24,14,7,8,5,2)\n");
for(i=0; i<n; i++)
{
scanf("%d", &data);
root=createtree(root, data);
}
break;
case 2: printf("\nInorder Traversal: \n");
inorder(root);
break;
case 3: printf("\nPreorder Traversal: \n");
preorder(root);
break;
case 4: printf("\nPostorder Traversal: \n");
postorder(root);
break;
case 5: exit(0);
default: printf("\nInvalid output");
break;
}}}
```

OUTPUT:

```
cc 3.c OR gcc 3.c
./a.out
```

```
1.Insertion in Binary Search Tree
2.Inorder
3.Preorder
4.Postorder
5.Exit
Enter your choice: 1
Enter N value: 12
Enter the values to create BST like(6,9,5,2,8,15,24,14,7,8,5,2)
6
9
5
2
8
15
24
14
7
8
5
2

1.Insertion in Binary Search Tree
2.Inorder
3.Preorder
4.Postorder
5.Exit
Enter your choice: 2
Inorder Traversal:
2 5 6 7 8 9 14 15 24

1.Insertion in Binary Search Tree
2.Inorder
3.Preorder
4.Postorder
5.Exit
Enter your choice: 3
Preorder Traversal:
6 5 2 9 8 7 15 14 24

1.Insertion in Binary Search Tree
2.Inorder
3.Preorder
4.Postorder
5.Exit
```

```
Enter your choice: 4
Postorder Traversal:
2 5 7 8 14 24 15 9 6

1.Insertion in Binary Search Tree
2.Inorder
3.Preorder
4.Postorder
5.Exit
Enter your choice: 5
```

5. **Design, Develop and implement a program in C for the following operations on Graph (G) of cities**
   a. Create a Graph of N cities using Adjacency Matrix.
   b. Print all the nodes reachable from a given starting node in a diagraph using DFS/BFS method.

```
#include<stdio.h>
#include<conio.h>
int a[10][10], n, m, i, j, source, s[10], b[10];
int visited[10];
void create()
{
printf("\nEnter the number of vertices of the digraph: ");
scanf("%d", &n);
printf("\nEnter the adjacency matrix of the graph:\n");
for(i=1; i<=n; i++)
for(j=1; j<=n; j++)
scanf("%d", &a[i][j]);
}
void bfs()
{
int q[10], u, front=0, rear=-1;
printf("\nEnter the source vertex to find other nodes reachable or not: ");
scanf("%d", &source);
q[++rear] = source;
visited[source] = 1;
printf("\nThe reachable vertices are: ");
while(front<=rear)
{
u = q[front++];
for(i=1; i<=n; i++)
{
if(a[u][i] == 1 && visited[i] == 0)
{
q[++rear] = i;
visited[i] = 1;
printf("\n%d", i);
}
}
}
}
void dfs(int source)
{
int v, top = -1;
s[++top] = 1;
b[source] = 1;
for(v=1; v<=n; v++)
{
if(a[source][v] == 1 && b[v] == 0)
{
```

```
printf("\n%d -> %d", source, v); dfs(v);
}
}
}
void main()
{
int ch;
while(1)
{
printf("\n1.Create Graph\n2.BFS\n3.Check graph connected or not(DFS)\n4.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)
{
case 1: create();
break;
case 2: bfs();
for(i=1;i<=n;i++)
if(visited[i]==0)
printf("\nThe vertex that is not rechable %d" ,i);
break;
case 3: printf("\nEnter the source vertex to find the connectivity: ");
scanf("%d",&source);
m=1;
dfs(source);
for(i=1;i<=n;i++)
{
if(b[i]==0)
m=0;
}
if(m==1)
printf("\nGraph is Connected");
else
printf("\nGraph is not Connected");
break;
default: exit(0);
}
}
}
```

OUTPUT:

```
cc 3.c OR gcc 3.c
./a.out
1. Create Graph
2.BFS
3.Check graph connected or not (DFS)
4.Exit
Enter your choice: 1
Enter the number of vertices of the digraph: 4
Enter the adjacency matrix of the graph:
0       0       1       1
0       0       0       0
0       1       0       0
0       1       0       0

1. Create Graph
2.BFS
3.Check graph connected or not (DFS)
4.Exit
Enter your choice: 2
Enter the source vertex to find other nodes reachable or not: 1
The reachable vertices are:
3
4
2

1. Create Graph
2.BFS
3.Check graph connected or not (DFS)
4.Exit
Enter your choice: 3
Enter the source vertex to find the connectivity: 1
1 -> 3
3 -> 2
1 -> 4
Graph is Connected

1. Create Graph
2.BFS
3.Check graph connected or not (DFS)
4.Exit
Enter your choice: 3
Enter the source vertex to find the connectivity: 2
Graph is not Connected

1. Create Graph
2.BFS
3.Check graph connected or not (DFS)
4.Exit
Enter your choice: 4
```

6. **Design, Develop and Implement a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)**
   a. Insert an Element on to Circular QUEUE
   b. Delete an Element from Circular QUEUE
   c. Demonstrate Overflow and Underflow situations on Circular QUEUE
   d. Display the status of Circular QUEUE
   e. Exit

```
#include<stdio.h>
#include<stdlib.h>
#define MAXSIZE 5
char cq[MAXSIZE];
int front, rear;
void insert(char item);
void del();
void display();
void main()
{
char item;
int choice, i;
front=-1;
rear=-1;
do
{
printf("\n\n...CIRCULAR QUEUE MENU...\n");
printf("\n 1.INSERT INTO QUEUE\n 2.DELETE FROM QUEUE\n 3.DISPLAY QUEUE\n
4.EXIT\n");
printf("\n\n ENTER YOUR CHOICE:");
scanf("%d", &choice);
switch(choice)
{
case 1:printf("\n ENTER THE CHARACTER input to the queue:");
scanf("%s",&item);
insert(item);
break;
case 2:del();
break;
case 3:display();
break;
case 4:exit(0);
default:printf("\n invalid choice\n");
}
}while(choice!=4);
}
void insert(char item)
{
if(front==(rear+1)%MAXSIZE)
printf("\n\n CIRCULAR QUEUE IS OVERFLOW\n");
else
{
if(front==-1)
front=rear=0;
else
rear=(rear+1)%MAXSIZE;
cq[rear]=item;
printf("\n rear=%d front=%d\n",rear,front);
}
}
void del()
{
char item;
if(front==-1)
printf("\n CIRCULAR QUEUE IS UNDERFLOW\n");
else
{
item=cq[front];
cq[front]='0';
}
if(front==rear)
front=rear=-1;
else
{
front=(front+1)%MAXSIZE;
printf("\n DELETED ELEMENT FROM QUEUE IS: %c\n", item);
printf("\n rear=%d front=%d\n",rear,front);
}
}
void display()
{
int i;
if(front==-1)
printf("CIRCULAR QUEUE IS EMPTY\n");
else
printf("the queue elements are\n");
for(i=0;i<=MAXSIZE;i++)
printf("%c\t",cq[i]);
}
```

OUTPUT:

```
cc 3.c OR gcc 3.c
./a.out
```

```
...CIRCULAR QUEUE MENU...
1.INSERT INTO QUEUE
2.DELETE FROM QUEUE
3.DISPLAY QUEUE
```

```
4.EXIT
ENTER YOUR CHOICE: 2
CIRCULAR QUEUE IS UNDERFLOW

...CIRCULAR QUEUE MENU...
1.INSERT INTO QUEUE
2.DELETE FROM QUEUE
3.DISPLAY QUEUE
4.EXIT
ENTER YOUR CHOICE: 3
CIRCULAR QUEUE IS EMPTY

...CIRCULAR QUEUE MENU...
1.INSERT INTO QUEUE
2.DELETE FROM QUEUE
3.DISPLAY QUEUE
4.EXIT
ENTER YOUR CHOICE: 1
ENTER THE CHARACTER input to the queue: A
rear=0 front=0

...CIRCULAR QUEUE MENU...
1.INSERT INTO QUEUE
2.DELETE FROM QUEUE
3.DISPLAY QUEUE
4.EXIT
ENTER YOUR CHOICE: 1
ENTER THE CHARACTER input to the queue: C
rear=1 front=0

...CIRCULAR QUEUE MENU...
1.INSERT INTO QUEUE
2.DELETE FROM QUEUE
3.DISPLAY QUEUE
4.EXIT
ENTER YOUR CHOICE: 1
ENTER THE CHARACTER input to the queue: M
rear=2 front=0

...CIRCULAR QUEUE MENU...
1.INSERT INTO QUEUE
2.DELETE FROM QUEUE
3.DISPLAY QUEUE
4.EXIT
ENTER YOUR CHOICE: 3
the queue elements are
A C M
```

```
...CIRCULAR QUEUE MENU...
1.INSERT INTO QUEUE
2.DELETE FROM QUEUE
3.DISPLAY QUEUE
4.EXIT
ENTER YOUR CHOICE: 1
ENTER THE CHARACTER input to the queue: S
rear=3 front=0

...CIRCULAR QUEUE MENU...
1.INSERT INTO QUEUE
2.DELETE FROM QUEUE
3.DISPLAY QUEUE
4.EXIT
ENTER YOUR CHOICE: 1
ENTER THE CHARACTER input to the queue: P
rear=4 front=0

...CIRCULAR QUEUE MENU...
1.INSERT INTO QUEUE
2.DELETE FROM QUEUE
3.DISPLAY QUEUE
4.EXIT
ENTER YOUR CHOICE: 1
ENTER THE CHARACTER input to the queue: E
CIRCULAR QUEUE IS OVERFLOW

...CIRCULAR QUEUE MENU...
1.INSERT INTO QUEUE
2.DELETE FROM QUEUE
3.DISPLAY QUEUE
4.EXIT
ENTER YOUR CHOICE: 3
the queue elements are
A C M S P

...CIRCULAR QUEUE MENU...
1.INSERT INTO QUEUE
2.DELETE FROM QUEUE
3.DISPLAY QUEUE
4.EXIT
ENTER YOUR CHOICE: 2
DELETED ELEMENT FROM QUEUE IS: A
rear=4 front=1

...CIRCULAR QUEUE MENU...
1.INSERT INTO QUEUE
2.DELETE FROM QUEUE
```

```
3.DISPLAY QUEUE
4.EXIT
ENTER YOUR CHOICE:1
ENTER THE CHARACTER input to the queue: V
rear=0 front=1

...CIRCULAR QUEUE MENU...
1.INSERT INTO QUEUE
2.DELETE FROM QUEUE
3.DISPLAY QUEUE
4.EXIT
ENTER YOUR CHOICE: 3
the queue elements are
V C M S P
.
..CIRCULAR QUEUE MENU...
1.INSERT INTO QUEUE
2.DELETE FROM QUEUE
3.DISPLAY QUEUE
4.EXIT
ENTER YOUR CHOICE: 4
```