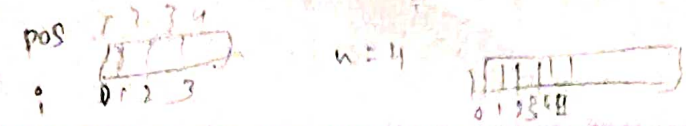
TERM WORK 1

→ Design, Develop & implement a menu driven program in C for following array operations.

i) Creating an array of n-integers elements
ii) Display of array elements with suitable headings
iii) Inserting a element at a given valid position
iv) Deleting an element at a given valid position
v) Exit.

```
#include <stdio.h>
#define MAX 20
  int main ()
  {
     int n, i, a[MAX], choice, x, pos;
     printf ("Enter the no. of elements");
     scanf ("%d", &n);
      for (i=0; i<n; i++)
       scanf ("%d", &a[MAX];

        do
        {
           printf ("MENU");
           printf ("1: Insert");
           printf ("2: Delete");
           printf ("3: Display");
           printf ("4: Exit");
            printf ("Enter your choice");
            scanf ("%d", &choice);

          switch (choice)
          {
             case 1: printf ("Enter the no. to
                              insert");
```

```
        scanf ("%d", &x);
        printf ("Enter the position to insert");
        scanf ("%d", &pos);
        for (i=n-1; i>=pos; i--)
            a[i+1] = a[i];
        a[pos] = x;
        n++;
        break;

case 2: printf ("Enter the position to delete");
        scanf ("%d", &pos);
        x = a[pos];
        for (i=pos+1; i<n; i++)
            a[i-1] = a[i];
        n--;

        printf ("Removed element is %d \n", x);
        break;

case 3: printf ("The element of array are");
        for (i=0; i<n; i++)
        {
            printf ("%d \n", a[i]);
    } //end switch

    } //end switch.

} while (choice<4);

} //end main.
```



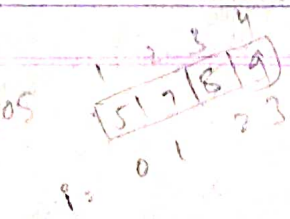
Output :-

Enter the no. of elements : 4
1
2
3
4

Enter your choice : 1 (insert)
Enter no. to insert : 2
Enter the position to insert : 2
Enter your choice : 8 (display)

The element of array are :

2
2
3
4.



Term work - 2

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

// Declarations.
char str[50], pat[20], rep[20], ans[50];
int i=0, j=0, c=0, m=0, k, flag=0;

void stringmatch()
{
  while(str[c]!='\0')
  {
    if(str[m]==pat[i])
    {
      i++;
      m++;
      if(pat[i]=='\0')
      {
        flag=1;
        for(k=0; rep[k]!='\0', k++, j++)
        {
          ans[j]=rep[k];
        }
        i=0;
        c=m;
      }
    }
    else
    {
      ans[j]=str[c];
      j++;
      c++;
```

```
            m=c;
            i=0;
        }
    }
        aus[j]='\0';
    }

void main().
{
    pf("\n Enter a main string\n");
    fgets(str, size of(str), stdin);
// gets(str);
printf("\n Enter a pattern string\n");
    fgets(pat, size of(pat), stdin);
// gets(pat);
    pf("\n Enter a replace string\n");
    fgets(rep, size of(rep), stdin);
    // get(rep);
        stringmatch();
        if(flag==1)
        pf("\n the resultant string is\n %os",
                                    aus);
        else
            pf("\n The pattern string not found");
```

Output:-

Enter the main string
This is mango.
Enter the pattern string
Mango
Enter replace string
Apple.
The resultant string is - This is apple.

Termwork - 3

Design, develop & implement a menu driven prog in C for the following operations on stack of Integers

a) push b) pop c) demonstrate on how stack can be used to check palindrome.

d) Demonstrate overflow & underflow

e) Display status of stack.

f) Exit.

```
#include <stdio.h>
int stk[6], rev[6];
int top = -1, k=0;
int size;
void push()
void pop()
void display()
int pali()
void main()
{
    int choice,i,;
    printf ("1: push\n 2: pop \n 3: display
    write()  4: Exit\n");
    scanf ("%d", &choice);
    switch (choice)
    {
        case1: push();
               break;
        case2: pop();
               break;
```

```
            case 3: display();
                    break;
            case 4: Exit();
                    break;
            }
        }
}.

void push()
{
    int num;
    if (top == -1)        if (top == size-1);
    printf ("stack overflow");
    else
    {
        printf ("Enter the element to push");
        scanf ("%d", &num);
        top++;
        stk [top] = num;
    }
void pop()
{
    int num;
    if (top == -1)
    printf ("underflow");
    else
    {
        num=stk [top];
        top--;
        printf ("The pop element is %d", num);
    }
void display()
{.
```

```
    int i;
    if (top==-1)
    printf ("stack is empty");
    else
    {
        printf ("contents are \n");
        for (i=top; i>=0; i--)
        printf ("%d", stk[i]);
    }
}
int pali()
{
    int i, flag=1;
    for (i=top; i>=0; i--)
    {
        if (stack[i] != rev[--k])
        {
            flag=0;
        }
    }
    return flag;
}
```

Output:-

Enter the size of stack

2

1: push \n 0: pop \n 3: display 4: Exit \n

Enter your choice

1

Enter the element to push

10

1: push \n 0: pop \n 3: display 4: Exit \n.

Enter your choice.

1

Enter the element to push.

10 20

2

20.

1: push\n 2: pop\n 3: Display\n 4: Exit\n

Enter your choice.

3.

The contents are.

10.

Termwork - 4.

Develop, Design & implement a menu driven program in C for converting an Infix Expression to postfix Expression. Program should support for both parentesize & free expression with the operators +, -, *, / & %

```
#include <stdio.h>
#include <ctype.h>
#define MAX 20
typedef struct
{
    char s[MAX];
    int top;
} stack;
stack stk;
void push (char);
char pop();
int isp (char);
int icp (char);
int icp (char symb);
{
    switch (symb)
    {
    case '(' : return 7;
    case '^' : return 6;
    case '*' : return 5;
    case '/' : return 6;
    case '%' : return 4;
    case '+' : return 3;
    case '-' : return 0;
```

```
        case '#': return -1;
    }
}

int isp (char symb)
{
    switch (symb)
    {
        case 'c' : return 0;
        case '^' : return 1;
        case '*' : return 3;
        case '/' : return 2;
        case '%' : return 4;
        case '+' : return 5;
        case '-' : return 6;
        case '#' : return -1;
    }
}

void push (char x)
{
    if (stk.top == MAX-1)
    printf ("overflow");
    else
    {
        stk.top++;
        stk.s [stk.top] = x;
    }
}

char pop ()
{
    if (stk.top == -1)
    printf ("underflow");
    else
    {
```

```
    char x;
    x = stk.[stk.top];
    stk.top--;
    return x;
  }
}
int main()
{
  char infix[MAX], symb;
  int i;
  stk.top = -1;
  printf("Enter the infix statement");
  scanf("%ds", infix);
  push('#');
  for(i=0; infix[i] != '\0'; i++)
  {
    symb = infix[i];
    if(isalnum(sym))
    printf("%c", symb);
    else if(symb == ')')
    {
      while(stk.s[stk.top] != '(')
      {
        printf("%c", pop());
      }
      stk.top--;
    }
    else
    {
      while(icp(symb) <= isp[stk.s[stk.top]])
      {
        printf("%c", pop());
      }
```

```
            push(symb);
        }
    }
    while (stk.s[stk.top] != '#')
    printf("%c", pop());
}
```

Output:-

Enter the infix statement-

a+b*c

abc*+

a+b*c+d-e.

abc*+d+e-.

Term work - 5

Design, develop & implement a program in C for the following stack application.

a) Evaluation of suffix expression with single digit operands & operators: +, -, *, /, %, ^

b) solving Towers of Hanoi problem with n

```
#include <stdio.h>
#include <math.h>
#include <ctype.h>
#define MAX 20

struct stack
{
    int s[MAX];
    int top;
} stk;

void push (int x)
{
    if (stk.top == MAX-1)
    printf ("stack is full");
    else
    {
        stk.top++;
        stk.s[stk.top] = x;
    }
}
int x;
if (stk.top == -1)
printf ("stack underflow");
else
{
```

```
    x = stk.s[stk.top];
    stk.top--;
    return(x);
}

int main()
{
    int i, opd2, opd1;
    char post[MAX], symb;
    stk.top = -1;
    printf("Enter the postfix expression");
    scanf("%s", post);
    for(i=0; post[i] != '\0'; i++)
    {
        symb = post[i];
        if(isdigit(symb))
            push(symb - '0')
        else
        {
            opd2 = pop();
            opd1 = pop();
            evaluate(opd1, opd2, symb);
        }
    }
    printf("Result = %d", pop());
}

void evaluate(int opd1, int opd2, char symb)
{
    int res;
    switch(symb)
    {
        case '+': res = opd1 + opd2;
                  push(res);
```

```
            break;
case '-' : res = opd1 - opd2
            push (res);
            break;
case '*' : res = opd1 * opd2
            push (res);
            break;
case '%' : res = opd1 % opd2;
            push (res);
            break;
case '/' : res = opd1 / opd2
            push (res);
            break;
case '^' : res = opd1 ^ opd2
            push (res);
            break;
        }
}
```

Output:-

ab * c +

a * b + c.

Termwork-6

Design Develop & implement a menu driven program in c for following operations on circular Queue of characters (Array implementation) of queue with maximum size (MAX)

```
#include <stdio.h>
#define MAX 20
  typedef struct
  {
      int a [MAX];
       int r, f;
  }q;
  void insert (q*);
  void remove (q*);
  void display (q);
   int main()
  {
      q q;
      int x, choice;
      q.f=-1, q.r=-1
      do
      {
         printf ("1: Insert\n 2: remove\n 3: display
                \n");
         printf ("Enter your choice")
         scanf ("%d", & choice);
         switch (choice)
         {
            case 1: insert (&q);
                    break;
```

```
case2: remove(&q);
       break;
case3: display(q);
       break;
printf("exit");
       break;

void insert(q* q)
{
   int prv;
   char x;
   prv = q->r;
   q->r = (q->r+1) % MAX;
   if (q->r == q->f)
   {
      printf("full");
      q->r = prev;
   }
   else
   {
      printf("Enter the character to insert");
      scanf("%c", &x);
      q->a[q->r) = x;
   }
}
void remove(q* q)
{
   char x;
   if (q->r == q->f)
   printf("empty");
   else
   {
```

```
q->f = (q->f + 1) % MAX;
x = q->a[q->f];
printf("%c", x);
void display(q q)
{
    int i;
    if (q.r == q.f)
    printf("empty");
    else if (q.f < q.r)
    {
        printf("queue elements are");
        for (i = q.f + 1; i <= q.r; i++)
        printf("%c", q.a[i]);
    }
    else
    {
        for(i = q.f + 1; i < MAX; i++)
        printf("%c", q.a[i]);
        for (i = 0; i <= q.r; i++);
        printf("%c", q.a[i]);
```

Output :-

Termwork - 8

Design, Develop & implement a menu driven progra
in C for the following operation on Doubly linke
list of Employee Data with the fields:
→ Create a DLL of N employee Data by using end
insertion.
Display the status of DLL & count the no. of
in it.
Perform insertion & deletion at end of DLL

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

    struct node
    {
        char name[20], dept[20], desg[20], sal
            ssn[20], phno[20];
            struct node *prv;
            struct node *next;
    } *first, *last, *temp;

    void create();
    void getnode();
    void insert_begin();
    void insert_end();
    void display();
    void displaynode(struct node *);

int main()
    {
        int choice;
```

```
    first = last = NULL;
    create ();
    do
    {
      printf ("\n\n MENU \n 1: insert begin
\n 2: insert end \n 3: delete ~~\n 4~~ begin
\n 4: delete end \n 5: display \n 6: exit");
      printf ("\n enter your choice :");
      scanf ("%d", &choice);
      switch (choice)
      {
        case 1: insert_begin();
                break;
        case 2: insert_end();
                break;
        case 3: delete_begin();
                break;
        case 4: delete_end();
                break;
        case 5: display();
                break;
      } while (choice < 6);
}

void create()
{
    int n, i;
    printf ("Enter no of elements :");
    scanf ("%d", &n);

    for (i=1 ; i<=n ; i++)
    {
      printf ("Enter details of student %d:
\n", i);
```

```
getnode ();

if (first == NULL)
first = last = temp;
else
{
last -> next = temp;
temp -> prv = last;
last = temp;
}
}
}

void getnode ()
{
temp = (struct node*) malloc (sizeof(
node));
printf ("\n enter name : ");
scanf ("%s", temp -> name);

printf ("\n enter dept : ");
scanf ("%s", temp -> dept);

printf ("\n enter desg : ");
scanf ("%s", temp -> desg);

printf ("\n enter sal : ");
scanf ("%s", temp -> sal);

printf ("\n enter ssn : ");
scanf ("%s", temp -> ssn);
```

Sini S

```
        printf ("\n enter phone number :");
        scanf ("%ld", temp -> phno);

        temp -> next = temp -> prv = NULL;
}

void insert_begin ()
{
    getnode ();
    if (first == NULL)
        first = last = temp;
    else
    {
        temp -> next = first;
        first -> prv = temp;
        first = temp;
    }
}

void insert_end ()
{
    getnode ();

    if (first == NULL)
        first = last = temp;
    else
    {
        last -> next = temp;
        temp -> prv = last;
        last = temp;
    }
}
```

```
void display node (struct node* temp)
{
    printf ("\n name : %s", temp->name);
    printf ("\n dept : %s", temp->dept);
    printf ("\n desg : %s", temp->desg);
    printf ("\n sal : %s", temp->sal);
    printf ("\n ssn : %s", temp->ssn);
    printf ("\n phone number : %s", temp->ph
}

void delete_begin().
{
    if (first = NULL)
    printf ("e \n empty list \n");
    else
    {
    if (first -> next == NULL)
    printf ("\n deleted employee details are:
        displaynode (first);
        free (first);
        first = last = NULL;
    }
    else
    {
        printf ("\n deleted employee detail
                    are : \n");
```

```
            displaynode (first);
            temp = first;
            first = first -> next
            first -> prv = NULL;
            free (temp);
        }
}

void delete_end ()
{
    if (first == NULL)
    printf ("e \n empty list \n");
    else
    if ( first -> next == NULL)
    {
        displaynode (first);
        free (first);
        first = last = NULL;
    }
else
{
    printf ("\n deleted employee details
                are : \n ");
    display node (last);
    temp = last;
    last = last -> prv;
    last -> next = NULL;
    free (temp);
}
}

void display ()
{
```

```
temp = first;
int i = 1;
if ( first == NULL)
printf ("\n nothing to display");
while (temp != NULL)
{
    printf ("\n employee %od dete
                are : \n", i );
    displaynode (temp);
    temp = temp -> next;
    i++;
}
}
```

Term work - 10.

- Design, Develop & 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 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 2

b) Traverse the BST in Inorder, Preorder & post order

c) Search the BST given element & repeat.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node
{
    struct node * lchild;
    struct node * rchild;
    int data;
}NODE;

NODE temp, prev, curr;
void pre (NODE root);
void in (NODE root);
void post(NODE root);
void insert (NODE * root, int num);
void search (NODE root, int key);

int main ()
{
    int choice, num, key;
    NODE root = NULL;

    do
    {
```

```
printf ("\MENU :\n");
printf ("1. Insert :\n");
printf ("2. Pre. order :\n");
printf ("3. Inorder :\n");
printf ("4' Postorder :\n");
printf ("5. search :\n");
printf ("6. Exit :\n");

printf ("Enter your choice\n");
scanf ("%d", &choice);
switch (choice).
{
case 1 : printf ("Enter no. to insert\n"
          scanf ("%d", &num);
          insert (&root, num);
          break ;

case 2 : pre (root);
          break;

case 3: in(root);
          break;

case 4 : post(root);
          break;

case 5 : printf ("Enter the element
                  to search\n");
          scanf ("%d", &key);
          search (root, key);
          break;
}
```

```
        }
        while (choice < 6);
}

void insert (NODE *root, int num).
{
    temp = (NODE) malloc (sizeof (struct node));
    temp -> data = num;
    temp -> lchild = NULL;
    temp -> rchild = NULL;

    if (*root == NULL).
        *root = temp;

    else
    {
        prev = NULL;
        cur = *root;
        while (cur != NULL)
        {
            prev = cur;
            if (num >= cur -> data)
                cur = cur -> rchild;
        }
        if (num >= prev -> data)
            prev -> rchild = temp;

        else.
            cur = cur -> lchild;
    }

void search (NODE root, int key).
```

```
{
  if (root == NULL)
  printf ("key not found \n");
  else
     if (root -> data == key)
     printf ("key found \n");
     else
        if (key >= root -> data)
        return search (root -> rchild, key);
        else.
        return search (root -> Lchild, key);
}.

void pre (NODE root)
{
     if (root != NULL)
     {
        printf ("%d", root -> data);
        pre (root -> Lchild);
        pre (root -> rchild);
     }
}

void in (NODE root)
{
     if (root != NULL)
     {
        in (root -> Lchild);
        printf ("%d", root -> data);
        in (root -> rchild);
     }
}
```

```
void post (NODE root)
{
    if (root != NULL)
    {
        post (root -> lchild);
        post (root -> rchild);
        printf ("%d", root -> data);
    }
}
```

Output :-

Preorder - 65 98788 15 14 24

Termwork - II

- Design, Develop & implement a program in C for the following operation on Graph (G) of cities.
  Create
  i) Graph of N citi using A adjacency matrix.
  ii) print all the nodes reachable from given starting node in a digraph using DFS/BFS method.

```
#include<stdio.h>
void bfs (int a[10][10], int v[], int, int);

int main()
{
    int a[10][10] = {0}, visit [10] = {0};
    int i, j, u, v, nodes, edges, start;
    printf ("Enter the no of nodes & edges \n");
    scanf ("%d %d", &edges, &nodes);
    printf ("Enter the edges \n");
    for (i=0; i< edges ; i++)
    {
        scanf ("%d %d", &u, &v);
        a[u][v] = 1;
    }

    printf ("Enter the starting node \n");
    scanf (" %d", &start);

    printf ("Reachable nodes are \n");
    bfs (a, visit, start, nodes);
}
```

```
void bfs (int a[10][10], int visit[10], int start,
    int nodes)
{
    int q[10], rear = -1, front = 0, i;
    q[++rear] = start;
    visit[start] = 1;
    while (front <= rear)
    {
        for (i = 0; i < nodes; i++)
        {
            if (visit[i] == 0 && a[start][i]
                                    == 1)
            {
                q[++rear] = i;
                visit[i] = 1;
            }
        }
        printf ("%d ", q[front++]);
        start = q[front];
    }
}
```

Output:-

Enter the number of nodes & edges.

8 10

Enter the edges

0
1
0
2
1
3

1
4
2
5
2
6
3
7
4
7
5
7
6
7

Enter the starting nodes.
0
Reachable nodes are.
0 1 2 3 4 5 6 7.