2. Design, Develop and Implement a Program in C for the following operationson Strings a. Read a main String (STR), a Pattern String (PAT) and a Replace String (REP) b. Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR. Report suitable messages in case PAT does not exist in STR Support the program with functions for each of the above operations. Don't use Built-in functions.

```
#include<stdio.h>
void main()
{
  char s[50],pat[20],rep[20],res[50];
  int i,j,k,l,flag;
  printf("\nEnte the main string:");
  gets(s);
  printf("\nEnter the pat string:");
  gets(pat);
  printf("\nEnter the replace string:");
  gets(rep);
  for(i=0,k=0;s[i]!='\0';i++)
  {
    flag=1;
    for(j=0;pat[j]!='\0';j++)
      if(s[i+j]!=pat[j])
        flag=0;
    l=j;
    if(flag)
    {
      for(j=0;rep[j]!='\0';j++,k++)
        res[k]=rep[j];
      i+=l-1;
    }
    else
      res[k++]=s[i];
  }
  res[k]='\0';
  printf("%s",res);
printf("\nthe string before pattern match is:\n %s",s);
if(flag==1)
printf("\nthe string after pattern match and replace is:\n %s",res);
else
printf("\npattern string is not found");
}
```

3.

```
#include <stdio.h>
#include <stdlib.h>
int s[5],top=-1;
void push()
{
   if(top==4)
      printf("\nStack overflow!!!!");
   else
   {
      printf("\nEnter element to insert:");
      scanf("%d",&s[++top]);
   }
}
void pop()
{
   if(top==-1)
      printf("\nStack underflow!!!");
   else
      printf("\nElement popped is: %d",s[top--]);
}
void disp()
{
   int t=top;
   if(t==-1)
      printf("\nStack empty!!");
   else
      printf("\nStack elements are:\n");
   while(t>=0)
      printf("%d ",s[t--]);
}
void pali()
{
   int num[5],rev[5],i,t;
   for(i=0,t=top;t>=0;i++,t--)
      num[i]=rev[t]=s[t];
   for(i=0;i<=top;i++)
      if(num[i]!=rev[i])
      break;
   /*printf(" num     rev\n");
   for(t=0;t<=top;t++)
     printf("%4d   %4d\n",num[t],rev[t]);*///remove /* */ to display num and rev
   if(i==top+1)
      printf("\nIt is a palindrome");
   else
      printf("\nIt is not a palindrome");
```

```
}
int main()
{
   int ch;
   do
   {
      printf("\n...Stack operations.....\n");
      printf("1.PUSH\n");
      printf("2.POP\n");
      printf("3.Palindrome\n");
      printf("4.Display\n");
      printf("5.Exit\n_____\n");
      printf("Enter choice:");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1:push();break;
         case 2:pop();break;
         case 3:pali();break;
         case 4:disp();break;
         case 5:exit(0);
         default:printf("\nInvalid choice");
      }
   }
   while(1);
   return 0;

}
```

4. Design, Develop and Implement a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, %(Remainder), ^(Power) and alphanumeric operands.

```
#include<stdio.h>
#include<string.h>
int F(char symbol)
{
 switch (symbol)
 {
 case '+':
 case '-':return 2;
 case '*':
 case '/':
 case '%':return 4;
 case '^':
 case '$':return 5;
 case '(':return 0;
 case '#':return -1;
 default :return 8;
 }
}
int G(char symbol)
{
 switch (symbol)
 {
 case '+':
 case '-':return 1;
 case '*':
 case '/':
 case '%':return 3;
 case '^':
 case '$':return 6;
 case '(':return 3;
 case ')':return 0;
 default :return 7;
 }
}

void infix_postfix(char infix[], char postfix[])
{
int top=-1, j=0, i;
char s[30], symbol;
s[++top] = '#';
for(i=0; i < strlen(infix); i++)
```

```
{
 symbol = infix[i];
 while (F(s[top]) > G(symbol))
 {
  postfix[j] = s[top--];
  j++;
 }
 if(F(s[top]) != G(symbol))
  s[++top] = symbol;
 else
  top--;
}
while(s[top] != '#')
 postfix[j++] = s[top--];
postfix[j] = '\0';
}
void main()
{
char infix[20], postfix[20];
printf("\nEnter a valid infix expression\n") ;
scanf ("%s", infix) ;
infix_postfix (infix, postfix);
printf("\nThe infix expression is:\n");
printf ("%s",infix);
printf("\nThe postfix expression is:\n");
printf ("%s",postfix) ;

}
```

5. Design, Develop and Implement a Program in C for the following Stack Applications
a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^

```
#include<stdio.h>
#include<math.h>
#include<string.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("\nEnter 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("\nThe result is : %f\n", res);

}
```

5. Design, Develop and Implement a Program in C for the following Stack Applications
b. Solving Tower of Hanoi problem with n disks

```
#include<stdio.h>
#include<math.h>
void tower(int n, int source, int temp, int destination);
void tower(int n, int source, int temp, int destination)
{
if(n == 0)
 return;
tower(n-1, source, destination, temp);
printf("\nMove disc %d from %c to %c", n, source, destination);
tower(n-1, temp, source, destination);
}
void main ()
{
int n;
printf("\nEnter the number of discs: \n\n");
scanf("%d", &n);
printf("\nThe sequence of moves involved in the Tower of Hanoi are\n");
tower(n, 'A', 'B', 'C');
printf("\n\nTotal Number of moves are: %d\n", (int)pow(2,n)-1);

}
```

6.

```
#include <stdio.h>
#include <stdlib.h>
#define max 5
int q[max],f=-1,r=-1;
void ins()
{
   if(f==(r+1)%max)
      printf("\nQueue overflow");
   else
   {
      if(f==-1)
         f++;
      r=(r+1)%max;
      printf("\nEnter element to be inserted:");
      scanf("%d",&q[r]);
   }
}
void del()
{
   if(r==-1)
      printf("\nQueue underflow");
   else
   {
      printf("\nElemnt deleted is:%d",q[f]);
      if(f==r)
         f=r=-1;
      else
         f=(f+1)%max;
   }
}
void disp()
{
   if(f==-1)
      printf("\nQueue empty");
   else
   {
      int i;
      printf("\nQueue elements are:\n");
      for(i=f;i!=r;i=(i+1)%max)
         printf("%d\t",q[i]);
      printf("%d",q[i]);
      printf("\nFront is at:%d\nRear is at:%d",q[f],q[r]);
   }
}
int main()
```

```
{
   printf("\nCircular Queue operations");
   printf("\n1.Insert");
   printf("\n2.Delete");
   printf("\n3.Display");
   printf("\n4.Exit");
   int ch;
   do{
      printf("\nEnter choice:");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1:ins();break;
         case 2:del();break;
         case 3:disp();break;
         case 4:exit(0);
         default:printf("\nInvalid choice...!");
      }
  }while(1);
   return 0;

}
```

11.

```
#include <stdio.h>
#include <stdlib.h>
int a[20][20],q[20],visited[20],reach[20],n,f=0,r=-1,count=0;
void bfs(int v)
{
   int i;
   for(i=1;i<=n;i++)
    if(a[v][i]&&!visited[i])
    {
     visited[i]=1;
     q[++r]=i;
    }
    if(f<=r)
     bfs(q[f++]);
}
void dfs(int v)
{
   int i;
   reach[v]=1;
   for(i=1;i<=n;i++)
    if(a[v][i]&&!reach[i])
    {
     printf("%d->%d\n",v,i);
     count++;
     dfs(i);
    }
}
int main()
{
   int v,ch,i,j;
   printf("\nenter no. of vertices:");
   scanf("%d",&n);
   for(i=1;i<=n;i++)
    reach[i]=visited[i]=q[i]=0;
   printf("\nEnter graph data in matrix form:\n");
   for(i=1;i<=n;i++)
    for(j=1;j<=n;j++)
     scanf("%d",&a[i][j]);
   printf("\n1.BFS\n2.DFS\n3.Exit\nEnter choice:");
   scanf("%d",&ch);
   switch(ch)
   {
      case 1:printf("\nEnter vertex:");
             scanf("%d",&v);
             bfs(v);
```

```
            printf("\nThe nodes that are reacheble from %d are:\n",v);
            for(i=1;i<=n;i++)
              if(visited[i])
                printf("%d ",i);
            break;
        case 2:dfs(1);
            if(count==n-1)
              printf("\ngraph is connected");
            else
              printf("\ngraph is not connected");
            break;
        case 3:exit(0);
        default:printf("\nInvalid choice");
    }
    return 0;
}
```

12.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
#define mod(x) x%MAX
void linear_prob(int a[],int num,int key){
   if(a[key]==-1)
      a[key]=num;
   else
   {
      printf("\nCollision detected!!");
      int i;
      for(i=mod(key+1);i!=key;i=mod(++i))
         if(a[i]==-1)
         break;
      if(i!=key)
      {
         printf("\nCollision avoided successfully\n");
         a[i]=num;   }
      else
         printf("\nHash table is full\n");  }
}
void display(int a[]){
   short ch,i;
   printf("\n1.Filtered display\n2.Display all\nEnter choice:");
   scanf("%d",&ch);
   printf("\nHash table is :\n");
   for(i=0;i<MAX;i++)
      if(a[i]>0||ch-1)
         printf("%d  %d\n",i,a[i]);}
int main(){
   int a[MAX],num,i;
   printf("\nCollision handling by linear probing");
   for(i=0;i<MAX;a[i++]=-1);
   do
   {
 printf("\nEnter the data:");
      scanf("%4d",&num);
      linear_prob(a,num,mod(num));
      printf("Do u wish to continue(1/0):");
      scanf("%d",&i);
   }while(i);
   display(a);
   return 0;

}
```

12.

```
#include<stdio.h>
#include<stdlib.h>
int key[20], n, m;
int * ht, index;
int count = 0;
void insert(int key) {
index = key % m;
while (ht[index] != -1) {
index = (index + 1) % m;
}
ht[index] = key;
count++;
}
void display() {
int i;
if (count == 0) {
printf("\nHash Table is empty");
return;
}
printf("\nHash Table contents are:\n ");
for (i = 0; i < m; i++)
printf("\n T[%d] --> %d ", i, ht[i]);
}
void main() {
int i;
printf("\nEnter the number of employee records (N) :");
scanf("%d", & n);printf("\nEnter the two digit memory
locations (m) for hash table:");
scanf("%d", & m);ht = (int * ) malloc(m * sizeof(int));
```

```
for (i = 0; i < m; i++)
ht[i] = -1;printf("\nEnter the four digit key values (K) for N
Employee Records:\n ");
for (i = 0; i < n; i++)
scanf("%d", & key[i]);for (i = 0; i < n; i++) {
if (count == m) {
printf("\n~~~Hash table is full. Cannot insert the record %d
key~~~", i + 1);
break;
}
insert(key[i]);
}//Displaying Keys inserted into hash table
display();
}
```