PROGRAM:1
```
#include<stdio.h>
#include<stdlib.h> #include<string.h>
struct Day  {  char *name; int date; char
*activity; };  struct Day create()  { struct
Day day;  day.name = (char *)malloc(20
* sizeof(char));  day.activity = (char *)
malloc(100 * sizeof(char)); printf("Enter
the day name: "); scanf("%s", day.
name);    printf("Enter the date: ");
scanf("%d", &day.date);    printf("Enter
the activity for the day: ");    scanf("
%[^\n]", day.activity);     return day;
void read(struct Day calendar[], int size)
{ for (int i = 0; i < size; i++) {    calendar[i]
= create();   } }    void display(struct Day
calendar[], int size)   {  printf("\nWeekly
Activity Details:\n"); for (int i = 0; i <
size; i++) {  printf("Day %d: %s\n", i + 1,
calendar[i].name);printf("Date:%d\n",
calendar[i].date);printf("Activity: %s\n",
calendar[i].activity);    printf("\n");   } }
int main()   {  int weekSize = 7;   struct
Day calendar[weekSize];
read(calendar, weekSize);
display(calendar, weekSize);
for (int i = 0; i < weekSize; i++) {
free(calendar[i].name);
free(calendar[i].activity);  }   return 0;  }
```

PROGRAM:2
```
#include<stdio.h>
char str[100], pat[50],rep[50],ans[100] ;
int i, j,c,m,k, flag=0;   void stringmatch()
{  i = m = c = j = 0;    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;
} } }   void main()   {  printf("\nEnter a
main string \n");      scanf("%s",str);
printf("\nEnter a pattern string \n");
scanf("%s",pat);      printf("\nEnter a
replace string \n"); scanf ("%s",rep);
stringmatch();    if(flag = = 1)
printf("\nThe resultant string is\n %s" ,
ans);   else     printf("\nPattern string
NOT found\n");   }
```

PROGRAM 5A:
```
#include <stdio.h>
#include<math.h>#include<ctype.h>
#include <string.h> double
compute(char symbol, double op1,
double 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() {
double 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[++to]= res ;    } }      res=s[top--];
printf("\nThe result is :%f\n",res);   }
```

PROGRAM 5B:
```
#include <stdio.h>
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");    scanf("%d", &n);    tower(n, 'A',
'B','C');      printf("\n\nTotal Number of
moves are: %d", (int)pow(2,n)-1);   }
```

PROGRAM:7
```
#include <stdio.h>
#include <string.h>     #define null 0
struct student     {    char usn[15],
name[20],branch[10];    int sem;   char
phno[20];     struct student *link; };
typedef struct student node;
node *start;      void main()    {   void
create(),insert_end(),del_front(),disp();
int ch;   while(1) { printf("Main Menu \n
");printf("1:Create\n2:Display\n3:Insert
Endt\n4:Delete Front\n5:Exit\n");
printf("Enter your choice\n");
scanf("%d",&ch);    switch(ch)   {
case 1:create(); break;     case 2:disp();
break;     case 3:insert_end(); break;
case 4:del_front(); break; case 5:exit(0);
} } }    void create()    { int i,n; node *p;
printf("Enter the number of students
\n");     scanf("%d",&n);     for(i=0;iusn,
p->name,p->branch,&p->sem,p->phno);
p->link=start; start=p;   } }    void disp() {
int cnt=0;  node *t;   t=start;     while(t) {
cnt++;     printf("%s\t%s\t%s\t%d\t%s-
>\n",t->usn,t->name,t->branch,t-
>sem,t->phno);   t=t->link; }
printf("Total number of nodes=% d\n\",
cnt);   }  void insert_end()  { node *p,*r;
p=(node*)malloc(sizeof(node));
printf("Enter the student USN , NAME
,BRANCH,SEM,PHNO\n");
scanf("%s%s%s%d%s",p->usn,p->name,
p->branch,&p->sem,p->phno);   r=start;
while(r->link!=null)  r=r->link;  r->link=p;
p->link=null;     }    void del_front()   {
node *q;     if(start==null)   {  printf("List
empty\n");     return;  }     q=start;
printf("Deleted node is %s",q->usn);
start=start->link;      free(q);     }
```

PROGRAM:6
```
#include <stdio.h>
#define MAX 4     int ch, front = 0, rear =
-1, count=0;     char q[MAX], item;  void
insert(char item)    {     if(count == MAX)
printf("\nQueue is Full");    return; else {
rear = (rear + 1) % MAX;    q[rear]=item;
count++;   } }  void del()   { if(count == 0)
printf("\nQueue is Empty");    return;
else  {   if(front > rear && rear==MAX-1)
{   front=0;  rear=-1;   count=0;  }   else {
item=q[front];    printf("\nDeleted item
is: %c",item);  front = (front + 1) % MAX;
count--;   } } }    void display()   {   int i,
f=front, r=rear;      if(count == 0)
printf("\nQueue is Empty");       else {
printf("\nContents of Queue is:\n");
for(i=0;i<=count;i++){printf("%c\t",q[f]);
f = (f + 1) % MAX;    } } }    void main()   {
do   {  printf("\n1. Insert\n2. Delete\n3.
Display\n4. Exit");    printf("\nEnter the
choice: ");  scanf("%d", &ch); switch(ch)
{   case 1:  printf("\nEnter the character
/ item to be inserted: ");      scanf("%c",
&item);    insert(item); break;      case 2:
del();   break;    case 3: display(); break;
case 4: exit(0); break;  } }while(ch!=4); }
```

PROGRAM:3
```
#include <stdio.h>
#include < string.h>     #include
<stdlib.h>    #define max 3
int st[max], top=-1;   void push(int item)
{   if(top==max-1)    {   printf("Stack
overflow\n");  return ; } st[++top]=item;
}    int pop()     { if(top==-1)   {
printf("Stack underflow\n"); return 0; }
return(st[top--]);    }   void palin() { int i,
len, count=0;     char p[100];   top=-1;
printf("Enter a string\n"); scanf("%s",p);
len=strlen(p);     for(i=0;i<len;i++) {
push(p[i]); }     for(i=0;i<len;i++) {
if(p[i]==pop())     {   count++;  }  }
if(len==count)   { printf("the string is
palindrome\n");   }   else { printf("the
string is not palindrome\n");  }}
 void disp()    {  int i;    if(top==-1)  {
printf("Stack Empty\n"); return; }
printf("the stack contents are");
for(i=top;i>=0;i--)
printf("|%d|\n",st[i]);    }  void main() {
int ch,k,item;   while(1)   {  printf("MAIN
MENU\n");   printf(" 1:Push\n 2:Pop\n
3:Display\n 4:Palindrome\n 5:Exit\n");
printf("Enter your choice\n");
scanf("%d",&ch);    switch(ch)   {
case 1:printf("Enter an item to push\n
");   scanf("%d",&item);   push(item);
break;    case 2: k=pop();    if(k)
printf("popped element is %d\n",k);
break;      case 3: disp(); break; case
4:palin();  break; case    5:exit(0);    }  }  }
```

PROGRAM:11
```
#include <stdio.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\n
2.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("\the vertex
that is not reachable %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("\n Graph is Connected");    else
printf("\n Graph is not Connected");
break; default: exit(0);      }  }  }
```

PROGRAM:4
```
#include <stdio.h>
#include <string.h>    int F(char symbol)
{   switch(symbol)   {   case '+' :   case '-':
return 2;    case '*':   case '/':  return 4;
case '^':    case '$':   return 5;   case '(':
return 0;    case '#':   return -1;   default:
return 8;   }   int  G(char symbol)   {
switch(symbol)    {    case '+':    case '-':
return 1;   case '*':    case '/':  return 3;
case '^':   case '$':  return 6;    case '(':
return 9;    case ')':   return 0;    default:
return 7;  } }    void infix_postfix(char
infix[], char postfix[])   {    int top, j, i;
char s[30], symbol;  top = -1;  s[++top] =
'#';  j = 0;  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);}
```

Program:8
```
#include <stdio.h> #include
<stdlib.h>  #define null 0    struct emp {
char name[40],dept[40],desig[40];
int ssn;    long int sal;     char phno[20];
struct emp *llink;    struct emp *rlink; };
typedef struct emp node;   node *start;
void create(),insert_front(),del_front(),
disp();    void main()  {    int ch; clrscr();
while(1)    {   printf("\nMain Menu\n");
printf("1:Create\n2:Display\n3:Insert_F
ront\n4:Del_Front\n5:Exit\n");
printf("Enter your choice\n");
scanf("%d",&ch);    switch(ch)    { case
1:create(); break;    case 2:disp(); break;
case 3:insert_front(); break;       case
4:del_front(); break; case  5:exit(0); } } }
void create()    {   node *p, *t;    int i, n;
printf("Enter the number of employees
\n");    scanf("%d", &n); printf("Enter
the employee details[SSN,NAME,DEPT,
DESIG,SAL AND PH.NO.]\n");   for(i=0;
irlink=null;    scanf("%d%s%s%s%ld%s",
&p->ssn,p-> name,p->dept,p->desig,
&p->sal,p->phno);     if(start==null)    {
start=p;    start->llink=null;    }    else {
t=start;    while(t->rlink!=null) t=t->rlink;
t->rlink=p;  p->llink=t;   } } } }  void disp()
{ node *r;    r=start;    while(r) {
printf("|%d|%s|%s|%s|%ld|%s|\n<-> "
, r->ssn,r->name,r->dept,r->desig,r->
sal, r->phno);      r=r->rlink;    } }
void insert_front()    {    node *p;
p=(node*)malloc(sizeof(node));
printf("Enter emp details\n");
scanf("%d%s%s%s%ld%%s", &p->ssn, p-
>name,p->dept, p->desig, &p->sal, p-
>phno);    p->rlink=start; start=p; start->
llink=null;   }  void del_front() { node *q;
if(start==null)   {   printf("list empty\n");
return; } q=start; printf("Deleted nodeis
%d",q->ssn);  start=start->rlink;free(q); }
```

```
PROGRAM:9    #include <stdio.h>
#include <stdlib.h>    #define
COMPARE(x,y)(((x)==(y))?0:((x)>(y))?1:-
1) struct node { int coeff; int expon;
struct node *link; }; typedef struct node
*NODE; NODE getnode()    {    NODE x;
x=(NODE) malloc(sizeof(struct node));
if(x==NULL)    {    printf("out of
memory");    exit(0);   }   return x;  }
NODE attach(int coeff,int expon,NODE
head)    {    NODE temp, cur;
temp=getnode();    temp-> coeff=coeff;
temp->expon=expon; cur=head->link;
while(cur-> link!=head)  { cur=cur->link;
}  cur-> link=temp;  temp->link=head;
return head;  }      NODE
read_poly(NODE head)    { int i=1; int
coeff; int expon; printf("enter the
coefficient as -999 to the end of the
polynomial"); while(1)    { printf("enter
the %d term\n",i++);    printf("Coeff=");
scanf("%d",&coeff);   if(coeff==-999)
break;     printf("pow x=");
scanf("%d",&expon);
head=attach(coeff,expon,head); }
return head; }
NODE poly_add(NODE head1,NODE
head2,NODE head3)   {   NODE a,b; int
coeff; a=head1->link;  b=head2-> link;
while(a!=head1 && b!=head2) {
switch(COMPARE(a->expon,b-> expon))
{   case 0: coeff=a->coeff+ b-> coeff;
if(coeff!=0)head3=attach(coeff,a->
expon,head3);  a=a->link; b=b->link;
break;    case 1: head3=attach(a->
coeff,a->expon,head3);  a=a->link;
break;   default: head3=attach(b->
coeff,b->expon,head3);   b=b->link; } }
while(a!=head1)   { head3=attach(a-
>coeff,a->expon, head3); a=a->link; }
while(b!=head2) { head3=attach(b-
>coeff,b->expon, head3);  b=b->link;  }
return head3; } void display(NODE
head)   { NODE temp;   if(head-
>link==head)  { printf("polynomial
doesnot exist"); return;   }
temp=head->link; while(temp!=head)  {
if(temp-> coeff <0)
printf("+%2dx^%2d",temp->  coeff
,temp->expon);    else
printf("+%2dx^%2d",temp-> coeff,
temp->expon);     temp=temp->link; } }
void main() {
NODE head1,head2,head3;
head1=getnode(); head2=getnode();
head3=getnode(); head1-> link=head1;
head2->link=head2; head3-
>link=head3;
printf("enter the first polynamial");
head1=read_poly(head1); printf("enter
the second polynomial ");
head2=read_poly(head2);
head3=poly_add(head1,head2,head3);
printf("polynomial1\n");
display(head1);
printf("\npolynomial2\n");
display(head2);
printf("\npolynomial3\n");
display(head3); }
```

```
PROGRAM:10    #include <stdio.h>
#include <stdlib.h>   struct BST   {   int
data;   struct BST *left;     struct BST
*right;   };     typedef struct BST *NODE;
NODE root;    NODE createtree(NODE
root, int data)    {    if (root == NULL) {
NODE temp;      temp= (NODE)malloc
(sizeof(NODE));      temp-> data = data;
temp->left = temp->right = NULL;
return temp; }   if (data < (root->data)) {
root->left = createtree(root->left, data);
}   else if (data > root->data)  {  root ->
right = createtree(root->right, data);  }
return root;    }    NODE search(int key
,NODE root)    {    if(root == NULL)
printf("\nElement not found");    else
if(key < root->data)    {   root->left =
search(key,root->left);   }   else   if(key >
root->data)  { root->right=search (key
,root->right);   }    else printf("\nElement
found is: %d", root->data);    return
root; }    void inorder(NODE root)  {
if(root != NULL)   { inorder(root->left);
printf("%d\t", root->data);
inorder(root->right);    } }    void
preorder(NODE root)   { if(root != NULL)
{    printf("%d\t", root->data);
preorder(root->left);   preorder(root->
right);    } }    void postorder(NODE
root)  { if(root != NULL) {
postorder(root->left);
postorder(root->right);   printf("%d\t",
root->data); } }    void main() { int data,
ch, i, n,key;    NODE *root=NULL;
while (1)    {    printf("\n1.Insertion
\n2.Inorder\n3.Preorder\n4.Postorder\
n5.search\n6.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 BST
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: printf("enetr the key element to
search\n"); scanf("%d",&key);
search(key,root);    break;
default:exit(0); }  }  }
```

```
PROGRAM:6       #include <stdio.h>
#define MAX 4     int ch, front = 0, rear =
-1, count=0;     char q[MAX], item;  void
insert(char item)     {     if(count == MAX)
printf("\nQueue is Full");     return; else {
rear = (rear + 1) % MAX;     q[rear]=item;
count++;   } }   void del()   { if(count == 0)
printf("\nQueue is Empty");     return;
else   {   if(front > rear && rear==MAX-1)
{   front=0;  rear=-1;   count=0;  }  else {
item=q[front];     printf("\nDeleted item
is: %c",item);  front = (front + 1) % MAX;
count--;   } } }     void display()   {   int i,
f=front, r=rear;      if(count == 0)
printf("\nQueue is Empty");       else {
printf("\nContents of Queue is:\n");
for(i=0;i<=count;i++){printf("%c\t",q[f]);
f = (f + 1) % MAX;     } } }     void main()   {
do   {  printf("\n1. Insert\n2. Delete\n3.
Display\n4. Exit");     printf("\nEnter the
choice: ");  scanf("%d", &ch); switch(ch)
{   case 1:  printf("\nEnter the character
/ item to be inserted: ");       scanf("%c",
&item);    insert(item); break;      case 2:
del();    break;    case 3: display(); break;
case 4: exit(0); break;  } }while(ch!=4); }
```

```
PROGRAM:11 #include <stdio.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\n
2.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("\the vertex
that is not reachable %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("\n Graph is Connected");    else
printf("\n Graph is not Connected");
break; default: exit(0);      }  }  }
```

```
Program:8  #include <stdio.h> #include
<stdlib.h>  #define null 0    struct emp {
char name[40],dept[40],desig[40];
int ssn;    long int sal;     char phno[20];
struct emp *llink;    struct emp *rlink; };
typedef struct emp node;    node *start;
void create(),insert_front(),del_front(),
disp();     void main()   {    int ch; clrscr();
while(1)    {    printf("\nMain Menu\n");
printf("1:Create\n2:Display\n3:Insert_F
ront\n4:Del_Front\n5:Exit\n");
printf("Enter your choice\n");
scanf("%d",&ch);    switch(ch)    { case
1:create(); break;    case 2:disp(); break;
case 3:insert_front(); break;        case
4:del_front(); break; case  5:exit(0); } } }
void create()    {    node *p, *t;    int i, n;
printf("Enter the number of employees
\n");    scanf("%d", &n); printf("Enter
the employee details[SSN,NAME,DEPT,
DESIG,SAL AND PH.NO.]\n");    for(i=0;
irlink=null;     scanf("%d%s%s%s%ld%s",
&p->ssn,p-> name,p->dept,p->desig,
&p->sal,p->phno);      if(start==null)    {
start=p;    start->llink=null;    }    else {
t=start;    while(t->rlink!=null) t=t->rlink;
t->rlink=p;  p->llink=t;    } } } }  void disp()
{ node *r;    r=start;    while(r) {
printf("|%d|%s|%s|%s|%ld|%s|\n<-> "
, r->ssn,r->name,r->dept,r->desig,r->
sal, r->phno);      r=r->rlink;    } }
void insert_front()     {      node *p;
p=(node*)malloc(sizeof(node));
printf("Enter emp details\n");
scanf("%d%s%s%s%ld%s", &p->ssn, p-
>name,p->dept, p->desig, &p->sal, p-
>phno);     p->rlink=start; start=p; start->
llink=null;   }  void del_front() { node *q;
if(start==null)    {    printf("list empty\n");
return; } q=start; printf("Deleted nodeis
%d",q->ssn);  start=start->rlink;free(q); }
```

```
PROGRAM:12   #include <stdio.h>
#include <stdlib.h>    #define MAX 10
struct employee   {   int id;   char
name[15];   };   typedef struct employee
EMP;    EMP emp[MAX];     int a[MAX];
int create(int num)    {    int key;   key =
num % 100;     return key;    }
int getemp(EMP emp[],int key)    {
printf("\nEnter emp id: ");
scanf("%d",&emp[key].id);
printf("\nEnter emp name: ");
scanf("%s",emp[key].name);     return
key;   }  void display()   {    int i, ch;
printf("\n1.Display ALL\n2.Filtered
Display");    printf("\nEnter the choice:
");     scanf("%d",&ch);    if(ch == 1)   {
printf("\nThe hash table is:\n");
printf("\nHTKey\tEmpID\tEmpName");
for(i=0; i<MAX; i++)
printf("\n%d\t%d\t%s", i, emp[i].id,
emp[i].name);  }      else   {
printf("\nThe hash table is:\n");
printf("\nHTKey\tEmpID\tEmpName");
for(i=0; i<MAX; i++)      if(a[i] != -1)   {
printf("\n%d\t%d\t%s", i, emp[i].id,
emp[i].name);   continue;    }   }   }
void linear_prob(int key, int num)    {
int flag, i, count = 0;    flag = 0;   if(a[key]
== -1)   {    a[key]=getemp(emp, key);   }
else   {     printf("\nCollision
Detected...!!!\n");     i = 0;   while(i <
MAX)    {    if (a[i] != -1)   {    count++;
break;  }   else   i++;   }
printf("\nCollision avoided successfully
using LINEAR PROBING\n");   if(count ==
MAX)    {   printf("\n Hash table is full");
display(emp);   exit(1);   }   else   {
getemp(emp,key+1);    }     for(i=key;
i<MAX; i++)   if(a[i] == -1)   {   a[i] = num;
flag = 1;   break;   }     i = 0;
while((i < key) && (flag == 0))   {   if(a[i]
== -1)  {   a[i] = num;   flag=1;   break;   }
i++;   }   }   }    void main()    {    int num,
key, i;   int ans = 1;    printf("\nCollision
handling by linear probing: ");   for (i=0;
i < MAX; i++)    {   a[i] = -1;   }   do    {
printf("\nEnter the data: ");
scanf("%d", &num);    key=create(num);
linear_prob(key,num);     printf("\nDo
you wish to continue? (1/0): ");
scanf("%d",&ans);     } while(ans);
display(emp);     }
```

```
PROGRAM:4    #include <stdio.h>
#include <string.h>    int F(char symbol)
{   switch(symbol)  {   case '+' :   case '-':
return 2;    case '*':   case '/':  return 4;
case '^':    case '$':  return 5;   case '(':
return 0;    case '#':   return -1;  default:
return 8;   }   int  G(char symbol)   {
switch(symbol)    {    case '+':    case '-':
return 1;   case '*':    case '/': return 3;
case '^':   case '$':  return 6;    case '(':
return 9;    case ')':   return 0;    default:
return 7;  } }    void infix_postfix(char
infix[], char postfix[])   {    int top, j, i;
char s[30], symbol;  top = -1;  s[++top] =
'#';  j = 0;  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);}
```

PROGRAM:1
```
#include<stdio.h>
#include<stdlib.h> #include<string.h>
struct Day  {  char *name; int date; char
*activity; };  struct Day create()  { struct
Day day;  day.name = (char *)malloc(20
* sizeof(char));  day.activity = (char *)
malloc(100 * sizeof(char)); printf("Enter
the day name: "); scanf("%s", day.
name);    printf("Enter the date: ");
scanf("%d", &day.date);    printf("Enter
the activity for the day: ");    scanf("
%[^\n]", day.activity);     return day;
void read(struct Day calendar[], int size)
{ for (int i = 0; i < size; i++) {    calendar[i]
= create();   } }    void display(struct Day
calendar[], int size)   {   printf("\nWeekly
Activity Details:\n"); for (int i = 0; i <
size; i++) {  printf("Day %d: %s\n", i + 1,
calendar[i].name);printf("Date:%d\n",
calendar[i].date);printf("Activity: %s\n",
calendar[i].activity);    printf("\n");   }  }
int main()   {  int weekSize = 7;   struct
Day calendar[weekSize];
read(calendar, weekSize);
display(calendar, weekSize);
for (int i = 0; i < weekSize; i++) {
free(calendar[i].name);
free(calendar[i].activity);  }   return 0;  }
```

PROGRAM:2
```
#include<stdio.h>
char str[100], pat[50],rep[50],ans[100] ;
int i, j,c,m,k, flag=0;   void stringmatch()
{  i = m = c = j = 0;    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;
} } }    void main()    {  printf("\nEnter a
main string \n");      scanf(“%s”,str);
printf("\nEnter a pattern string \n");
scanf(“%s”,pat);       printf("\nEnter a
replace string \n"); scanf (“%s”,rep);
stringmatch();     if(flag = = 1)
printf("\nThe resultant string is\n %s" ,
ans);    else     printf("\nPattern string
NOT found\n");   }
```

PROGRAM 5A:
```
#include <stdio.h>
#include<math.h>#include<ctype.h>
#include <string.h> double
compute(char symbol, double op1,
double 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() {
double 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[++to]= res ;    }  }      res=s[top--];
printf(“\nThe result is :%f\n”,res);   }
```

PROGRAM 5B:
```
#include <stdio.h>
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");     scanf("%d", &n);     tower(n, 'A',
'B','C');       printf("\n\nTotal Number of
moves are: %d", (int)pow(2,n)-1);    }
```

PROGRAM:7
```
#include <stdio.h>
#include <string.h>     #define null 0
struct student     {    char usn[15],
name[20],branch[10];    int sem;   char
phno[20];     struct student *link; };
typedef struct student node;
node *start;      void main()    {   void
create(),insert_end(),del_front(),disp();
int ch;   while(1) { printf("Main Menu \n
");printf("1:Create\n2:Display\n3:Insert
Endt\n4:Delete Front\n5:Exit\n");
printf("Enter your choice\n");
scanf("%d",&ch);    switch(ch)   {
case 1:create(); break;     case 2:disp();
break;    case 3:insert_end(); break;
case 4:del_front(); break; case 5:exit(0);
} } }    void create()    { int i,n; node *p;
printf("Enter the number of students
\n");    scanf("%d",&n);    for(i=0;iusn,
p->name,p->branch,&p->sem,p->phno);
p->link=start; start=p;   } }     void disp() {
int cnt=0;  node *t;   t=start;     while(t) {
cnt++;     printf("%s\t%s\t%s\t%d\t%s-
>\n",t->usn,t->name,t->branch,t-
>sem,t->phno);    t=t->link; }
printf("Total number of nodes=% d\n\",
cnt);   }   void insert_end()  { node *p,*r;
p=(node*)malloc(sizeof(node));
printf("Enter the student USN , NAME
,BRANCH,SEM,PHNO\n");
scanf("%s%s%s%d%s",p->usn,p->name,
p->branch,&p->sem,p->phno);    r=start;
while(r->link!=null)  r=r->link;  r->link=p;
p->link=null;     }     void del_front()    {
node *q;     if(start==null)  {  printf("List
empty\n");     return;   }      q=start;
printf("Deleted node is %s",q->usn);
start=start->link;      free(q);       }
```

PROGRAM:6
```
#include <stdio.h>
#define MAX 4     int ch, front = 0, rear =
-1, count=0;     char q[MAX], item;  void
insert(char item)     {     if(count == MAX)
printf("\nQueue is Full");    return; else {
rear = (rear + 1) % MAX;    q[rear]=item;
count++;   } }   void del()   { if(count == 0)
printf("\nQueue is Empty");     return;
else {    if(front > rear && rear==MAX-1)
{   front=0;  rear=-1;   count=0;  }    else {
item=q[front];     printf("\nDeleted item
is: %c",item);  front = (front + 1) % MAX;
count--;   } } }     void display()   {   int i,
f=front, r=rear;      if(count == 0)
printf("\nQueue is Empty");       else {
printf("\nContents of Queue is:\n");
for(i=0;i<=count;i++){printf("%c\t",q[f]);
f = (f + 1) % MAX;    } } }     void main()   {
do   {  printf("\n1. Insert\n2. Delete\n3.
Display\n4. Exit");    printf("\nEnter the
choice: ");  scanf("%d", &ch); switch(ch)
{   case 1:  printf("\nEnter the character
/ item to be inserted: ");      scanf("%c",
&item);    insert(item); break;     case 2:
del();    break;    case 3: display(); break;
case 4: exit(0); break;  } }while(ch!=4); }
```

PROGRAM:3
```
#include <stdio.h>
#include < string.h>      #include
<stdlib.h>     #define max 3
int st[max], top=-1;   void push(int item)
{   if(top==max-1)    {   printf("Stack
overflow\n");  return ; } st[++top]=item;
}    int pop()      { if(top==-1)   {
printf("Stack underflow\n"); return 0; }
return(st[top--]);    }    void palin() { int i,
len, count=0;     char p[100];   top=-1;
printf("Enter a string\n"); scanf("%s",p);
len=strlen(p);     for(i=0;i<len;i++) {
push(p[i]); }      for(i=0;i<len;i++) {
if(p[i]==pop())     {   count++;  }  }
if(len==count)   { printf(“the string is
palindrome\n”);   }   else { printf(“the
string is not palindrome\n”);  }}
 void disp()    {  int i;    if(top==-1)  {
printf(“Stack Empty\n”); return; }
printf(“the stack contents are”);
for(i=top;i>=0;i--)
printf("|%d|\n",st[i]);    }  void main() {
int ch,k,item;   while(1)   {  printf("MAIN
MENU\n");   printf(" 1:Push\n 2:Pop\n
3:Display\n 4:Palindrome\n 5:Exit\n");
printf("Enter your choice\n");
scanf("%d",&ch);    switch(ch)   {
case 1:printf("Enter an item to push\n
");    scanf("%d",&item);    push(item);
break;     case 2: k=pop();    if(k)
printf("popped element is %d\n",k);
break;        case 3: disp(); break; case
4:palin();  break; case    5:exit(0);    }  }  }
```

PROGRAM:11
```
#include <stdio.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\n
2.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("\the vertex
that is not reachable %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("\n Graph is Connected");     else
printf("\n Graph is not Connected");
break; default: exit(0);       }  }  }
```

PROGRAM:4
```
#include <stdio.h>
#include <string.h>     int F(char symbol)
{   switch(symbol)   {   case '+' :   case '-':
return 2;    case '*':   case '/':  return 4;
case '^':    case '$':   return 5;   case '(':
return 0;    case '#':    return -1;   default:
return 8;   }   int  G(char symbol)   {
switch(symbol)    {    case '+':    case '-':
return 1;   case '*':    case '/':  return 3;
case '^':   case '$':  return 6;    case '(':
return 9;    case ')':   return 0;    default:
return 7;  } }     void infix_postfix(char
infix[], char postfix[])    {    int top, j, i;
char s[30], symbol;  top = -1;  s[++top] =
'#';  j = 0;  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);}
```

Program:8
```
#include <stdio.h> #include
<stdlib.h>  #define null 0    struct emp {
char name[40],dept[40],desig[40];
int ssn;    long int sal;     char phno[20];
struct emp *llink;    struct emp *rlink; };
typedef struct emp node;    node *start;
void create(),insert_front(),del_front(),
disp();     void main()  {     int ch; clrscr();
while(1)    {    printf("\nMain Menu\n");
printf("1:Create\n2:Display\n3:Insert_F
ront\n4:Del_Front\n5:Exit\n");
printf("Enter your choice\n");
scanf("%d",&ch);    switch(ch)    { case
1:create(); break;    case 2:disp(); break;
case 3:insert_front(); break;       case
4:del_front(); break; case  5:exit(0); } } }
void create()    {   node *p, *t;    int i, n;
printf("Enter the number of employees
\n");    scanf("%d", &n); printf("Enter
the employee details[SSN,NAME,DEPT,
DESIG,SAL AND PH.NO.]\n");    for(i=0;
irlink=null;      scanf("%d%s%s%s%ld%s",
&p->ssn,p-> name,p->dept,p->desig,
&p->sal,p->phno);       if(start==null)    {
start=p;    start->llink=null;    }    else {
t=start;    while(t->rlink!=null) t=t->rlink;
t->rlink=p;  p->llink=t;   } } } }  void disp()
{ node *r;    r=start;    while(r) {
printf("|%d|%s|%s|%s|%ld|%s|\n<-> "
, r->ssn,r->name,r->dept,r->desig,r->
sal, r->phno);      r=r->rlink;    } }
void insert_front()    {     node *p;
p=(node*)malloc(sizeof(node));
printf("Enter emp details\n");
scanf("%d%s%s%s%ld%s", &p->ssn, p-
>name,p->dept, p->desig, &p->sal, p-
>phno);    p->rlink=start; start=p; start->
llink=null;   }  void del_front() { node *q;
if(start==null)   {   printf("list empty\n");
return; } q=start; printf("Deleted nodeis
%d",q->ssn);  start=start->rlink;free(q); }
```

PROGRAM:9
```
#include <stdio.h>
#include <stdlib.h>    #define
COMPARE(x,y)(((x)==(y))?0:((x)>(y))?1:-
1) struct node { int coeff; int expon;
struct node *link; }; typedef struct node
*NODE; NODE getnode()   {   NODE x;
x=(NODE) malloc(sizeof(struct node));
if(x==NULL)   {   printf("out of
memory");   exit(0);   }   return x;  }
NODE attach(int coeff,int expon,NODE
head)   {   NODE temp, cur;
temp=getnode();   temp-> coeff=coeff;
temp->expon=expon; cur=head->link;
while(cur-> link!=head)  {  cur=cur->link;
}  cur-> link=temp;  temp->link=head;
return head;   }     NODE
read_poly(NODE head)   { int i=1; int
coeff; int expon; printf("enter the
coefficient as -999 to the end of the
polynomial"); while(1)   { printf("enter
the %d term\n",i++);   printf("Coeff=");
scanf("%d",&coeff);   if(coeff==-999)
break;    printf("pow x=");
scanf("%d",&expon);
head=attach(coeff,expon,head); }
return head; }
NODE poly_add(NODE head1,NODE
head2,NODE head3)   {   NODE a,b; int
coeff; a=head1->link;  b=head2-> link;
while(a!=head1 && b!=head2) {
switch(COMPARE(a->expon,b-> expon))
{   case 0: coeff=a->coeff+ b-> coeff;
if(coeff!=0)head3=attach(coeff,a->
expon,head3);  a=a->link; b=b->link;
break;   case 1: head3=attach(a->
coeff,a->expon,head3);  a=a->link;
break;   default: head3=attach(b->
coeff,b->expon,head3);   b=b->link; } }
while(a!=head1)   { head3=attach(a-
>coeff,a->expon, head3); a=a->link; }
while(b!=head2) { head3=attach(b-
>coeff,b->expon, head3);  b=b->link;  }
return head3; } void display(NODE
head)   { NODE temp;   if(head-
>link==head)  { printf("polynomial
doesnot exist"); return;   }
temp=head->link; while(temp!=head)  {
if(temp-> coeff <0)
printf("+%2dx^%2d",temp->  coeff
,temp->expon);    else
printf("+%2dx^%2d",temp-> coeff,
temp->expon);    temp=temp->link; } }
void main() {
NODE head1,head2,head3;
head1=getnode(); head2=getnode();
head3=getnode(); head1-> link=head1;
head2->link=head2; head3-
>link=head3;
printf("enter the first polynamial");
head1=read_poly(head1); printf("enter
the second polynomial ");
head2=read_poly(head2);
head3=poly_add(head1,head2,head3);
printf("polynomial1\n");
display(head1);
printf("\npolynomial2\n");
display(head2);
printf("\npolynomial3\n");
display(head3); }
```

PROGRAM:10
```
#include <stdio.h>
#include <stdlib.h>   struct BST   {   int
data;   struct BST *left;     struct BST
*right;   };     typedef struct BST *NODE;
NODE root;    NODE createtree(NODE
root, int data)   {    if (root == NULL) {
NODE temp;      temp= (NODE)malloc
(sizeof(NODE));      temp-> data = data;
temp->left = temp->right = NULL;
return temp; }   if (data < (root->data)) {
root->left = createtree(root->left, data);
}   else if (data > root->data)  {  root ->
right = createtree(root->right, data);  }
return root;   }     NODE search(int key
,NODE root)   {     if(root == NULL)
printf("\nElement not found");     else
if(key < root->data)    {    root->left =
search(key,root->left);   }   else   if(key >
root->data)  { root->right=search (key
,root->right);   }    else printf("\nElement
found is: %d", root->data);    return
root; }    void inorder(NODE root)  {
if(root != NULL)   { inorder(root->left);
printf("%d\t", root->data);
inorder(root->right);    } }    void
preorder(NODE root)   { if(root != NULL)
{    printf("%d\t", root->data);
preorder(root->left);   preorder(root->
right);    } }     void postorder(NODE
root)  { if(root != NULL) {
postorder(root->left);
postorder(root->right);   printf("%d\t",
root->data); } }    void main() { int data,
ch, i, n,key;    NODE *root=NULL;
while (1)    {    printf("\n1.Insertion
\n2.Inorder\n3.Preorder\n4.Postorder\
n5.search\n6.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 BST
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: printf("enetr the key element to
search\n"); scanf("%d",&key);
search(key,root);    break;
default:exit(0); }  }  }
```

PROGRAM:6
```
#include <stdio.h>
#define MAX 4    int ch, front = 0, rear =
-1, count=0;    char q[MAX], item;  void
insert(char item)    {     if(count == MAX)
printf("\nQueue is Full");    return; else {
rear = (rear + 1) % MAX;    q[rear]=item;
count++;   } }  void del()   { if(count == 0)
printf("\nQueue is Empty");    return;
else  {   if(front > rear && rear==MAX-1)
{   front=0;  rear=-1;  count=0;  }  else {
item=q[front];    printf("\nDeleted item
is: %c",item);  front = (front + 1) % MAX;
count--;   } } }    void display()   {   int i,
f=front, r=rear;      if(count == 0)
printf("\nQueue is Empty");      else {
printf("\nContents of Queue is:\n");
for(i=0;i<=count;i++){printf("%c\t",q[f]);
f = (f + 1) % MAX;    } } }     void main()   {
do   {  printf("\n1. Insert\n2. Delete\n3.
Display\n4. Exit");    printf("\nEnter the
choice: ");  scanf("%d", &ch); switch(ch)
{   case 1:  printf("\nEnter the character
/ item to be inserted: ");      scanf("%c",
&item);    insert(item); break;      case 2:
del();    break;    case 3: display(); break;
case 4: exit(0); break;  } }while(ch!=4); }
```

PROGRAM:11
```
#include <stdio.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\n
2.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("\the vertex
that is not reachable %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("\n Graph is Connected");     else
printf("\n Graph is not Connected");
break; default: exit(0);      }  }  }
```

Program:8
```
#include <stdio.h> #include
<stdlib.h>  #define null 0    struct emp {
char name[40],dept[40],desig[40];
int ssn;    long int sal;     char phno[20];
struct emp *llink;    struct emp *rlink; };
typedef struct emp node;    node *start;
void create(),insert_front(),del_front(),
disp();    void main()   {    int ch; clrscr();
while(1)    {    printf("\nMain Menu\n");
printf("1:Create\n2:Display\n3:Insert_F
ront\n4:Del_Front\n5:Exit\n");
printf("Enter your choice\n");
scanf("%d",&ch);    switch(ch)    { case
1:create(); break;    case 2:disp(); break;
case 3:insert_front(); break;        case
4:del_front(); break; case  5:exit(0); } } }
void create()   {    node *p, *t;    int i, n;
printf("Enter the number of employees
\n");    scanf("%d", &n); printf("Enter
the employee details[SSN,NAME,DEPT,
DESIG,SAL AND PH.NO.]\n");    for(i=0;
irlink=null;     scanf("%d%s%s%s%ld%s",
&p->ssn,p-> name,p->dept,p->desig,
&p->sal,p->phno);       if(start==null)    {
start=p;    start->llink=null;    }    else {
t=start;    while(t->rlink!=null) t=t->rlink;
t->rlink=p;   p->llink=t;   } } } }  void disp()
{ node *r;    r=start;    while(r) {
printf("|%d|%s|%s|%s|%ld|%s|\n<-> "
, r->ssn,r->name,r->dept,r->desig,r->
sal, r->phno);      r=r->rlink;    } }
void insert_front()     {      node *p;
p=(node*)malloc(sizeof(node));
printf("Enter emp details\n");
scanf("%d%s%s%s%ld%s", &p->ssn, p-
>name,p->dept, p->desig, &p->sal, p-
>phno);     p->rlink=start; start=p; start->
llink=null;   }  void del_front() { node *q;
if(start==null)    {    printf("list empty\n");
return; } q=start; printf("Deleted nodeis
%d",q->ssn);  start=start->rlink;free(q); }
```

PROGRAM:12
```
#include <stdio.h>
#include <stdlib.h>    #define MAX 10
struct employee   {    int id;    char
name[15];  };   typedef struct employee
EMP;     EMP emp[MAX];     int a[MAX];
int create(int num)    {     int key;    key =
num % 100;      return key;    }
int getemp(EMP emp[],int key)    {
printf("\nEnter emp id: ");
scanf("%d",&emp[key].id);
printf("\nEnter emp name: ");
scanf("%s",emp[key].name);      return
key;   }  void display()   {     int i, ch;
printf("\n1.Display ALL\n2.Filtered
Display");     printf("\nEnter the choice:
");     scanf("%d",&ch);     if(ch == 1)    {
printf("\nThe hash table is:\n");
printf("\nHTKey\tEmpID\tEmpName");
for(i=0; i<MAX; i++)
printf("\n%d\t%d\t%s", i, emp[i].id,
emp[i].name);   }       else    {
printf("\nThe hash table is:\n");
printf("\nHTKey\tEmpID\tEmpName");
for(i=0; i<MAX; i++)      if(a[i] != -1)   {
printf("\n%d\t%d\t%s", i, emp[i].id,
emp[i].name);   continue;    }   }   }
void linear_prob(int key, int num)    {
int flag, i, count = 0;    flag = 0;   if(a[key]
== -1)   {    a[key]=getemp(emp, key);   }
else    {     printf("\nCollision
Detected...!!!\n");     i = 0;    while(i <
MAX)     {     if (a[i] != -1)    {    count++;
break;  }   else   i++;    }
printf("\nCollision avoided successfully
using LINEAR PROBING\n");  if(count ==
MAX)    {   printf("\n Hash table is full");
display(emp);   exit(1);    }   else    {
getemp(emp,key+1);   }     for(i=key;
i<MAX; i++)   if(a[i] == -1)   {   a[i] = num;
flag = 1;   break;    }     i = 0;
while((i < key) && (flag == 0))   {   if(a[i]
== -1)   {   a[i] = num;   flag=1;   break;   }
i++;   }   }   }    void main()    {   int num,
key, i;   int ans = 1;    printf("\nCollision
handling by linear probing: ");    for (i=0;
i < MAX; i++)    {   a[i] = -1;   }   do    {
printf("\nEnter the data: ");
scanf("%d", &num);     key=create(num);
linear_prob(key,num);      printf("\nDo
you wish to continue? (1/0): ");
scanf("%d",&ans);     }  while(ans);
display(emp);     }
```

PROGRAM:4
```
#include <stdio.h>
#include <string.h>    int F(char symbol)
{   switch(symbol)   {   case '+' :   case '-':
return 2;    case '*':   case '/':  return 4;
case '^':    case '$':  return 5;   case '(':
return 0;   case '#':   return -1;   default:
return 8;   }   int  G(char symbol)   {
switch(symbol)    {    case '+':    case '-':
return 1;   case '*':    case '/':  return 3;
case '^':    case '$':  return 6;    case '(':
return 9;    case ')':   return 0;    default:
return 7;   } }    void infix_postfix(char
infix[], char postfix[])   {    int top, j, i;
char s[30], symbol;  top = -1;  s[++top] =
'#';  j = 0;  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);}
```