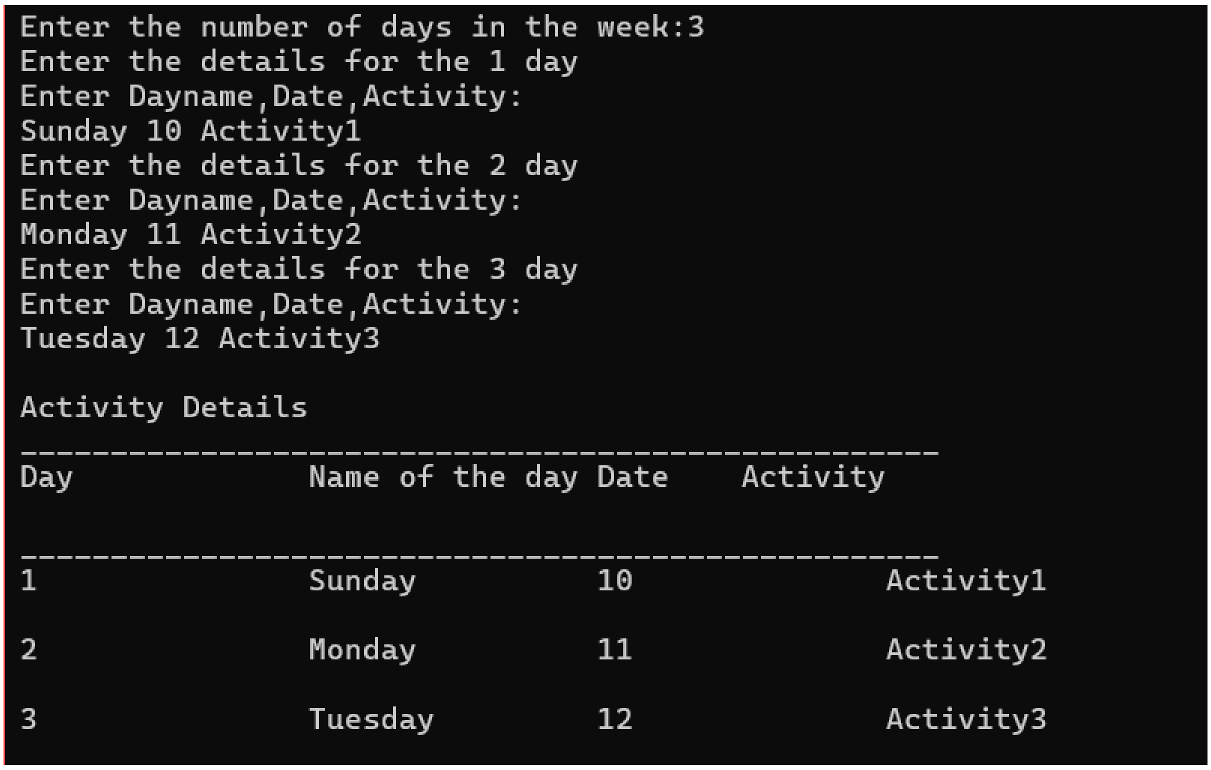
PROGRAM 1

1. Develop a menu driven Program in C for the following Array operations :
a. Declare a Calendar as an array of 7 elements (A dynamically Created array) to represent 7 days of a week. Each Element of the array is a structure having three fields. The first field is the name of the Day (A dynamically allocated string), The second field is the date of the Day (A integer), the third field is the description of the activity for a particular day (A dynamically allocated String).
b. Write functions create(), read(), and display(); to create the calendar, to read the data from the keyboard and to print weeks activity details report on screen.

```
#include<stdio.h>
#include<stdlib.h>
struct day {
  char *dayname;
  int date;
  char *activity;
};
typedef struct day DAY;

void create(DAY *day){
  day->dayname = (char*)malloc(20*sizeof(char));
  day->activity = (char*)malloc(50*sizeof(char));
  printf("\nEnter Dayname,Date,Activity:\n");
  scanf("%s%d%s",day->dayname,&day->date,day->activity);
}

void read(DAY *day , int size){
    for(int i=0;i<size;i++){
        printf("Enter the details for the %d day",i+1);
        create(&day[i]);
    }
}

void freememory(DAY *day,int size){
    for(int i=0;i<size;i++){
        free(day[i].dayname);
        free(day[i].activity);
    }
}

void display(DAY *day,int size){
  printf("\nActivity Details");
  printf("\n_______________________________________________________");
  printf("\nDay\t\tName of the day\tDate\tActivity\n");
  printf("\n_____________________________________________________");

    for(int i = 0;i<size;i++){
        printf("\n%d\t\t%s\t\t%d\t\t%s\n",i+1,day[i].dayname,day[i].date,day[i].activity);
    }
}
int main(){
  int n;
  printf("Enter the number of days in the week:");
  scanf("%d",&n);

  DAY *calender;
  calender = (DAY*)malloc(n*sizeof(DAY));
  if(calender==NULL){
    printf("Memory Allocation Falied!!");
    return 1;
  }
  read(calender,n);
  display(calender,n);
  freememory(calender,n);
  free(calender);
  return 0;
}
```

OUTPUT



```
Enter the number of days in the week:3
Enter the details for the 1 day
Enter Dayname,Date,Activity:
Sunday 10 Activity1
Enter the details for the 2 day
Enter Dayname,Date,Activity:
Monday 11 Activity2
Enter the details for the 3 day
Enter Dayname,Date,Activity:
Tuesday 12 Activity3

Activity Details
---------------------------------------------------
Day             Name of the day Date    Activity


---------------------------------------------------
1               Sunday          10              Activity1

2               Monday          11              Activity2

3               Tuesday         12              Activity3
```

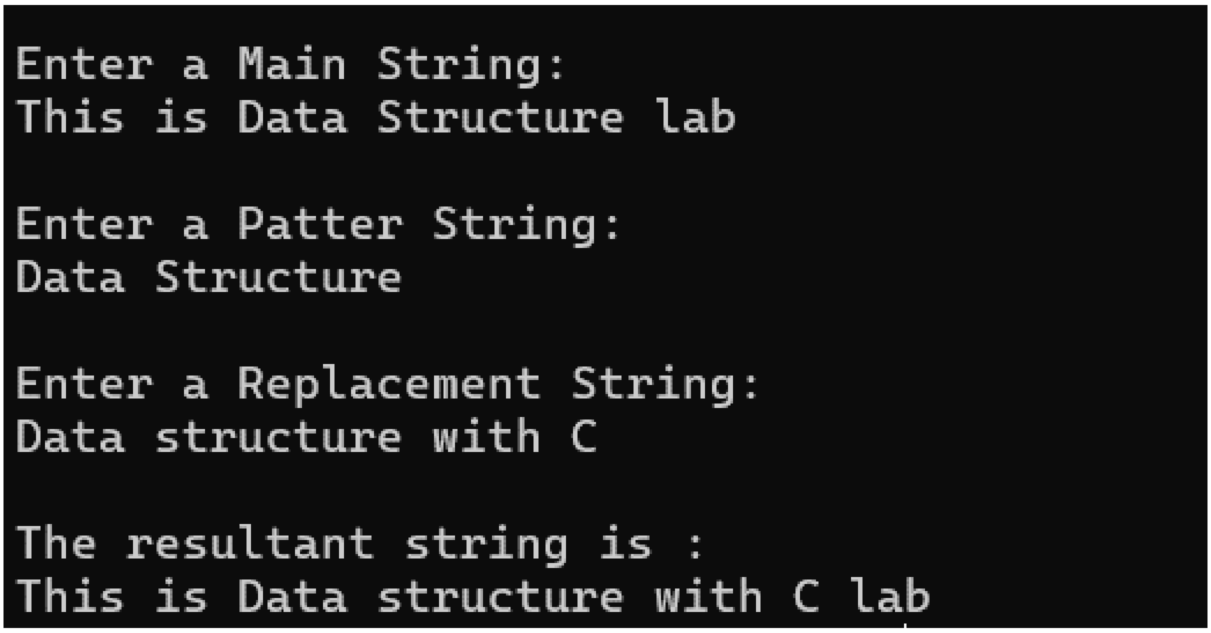
PROGRAM 2

2. Develop a Program in C for the following operations on 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>
#include<stdlib.h>
#include<string.h>
char str[50],pat[50],rep[50],ans[50];
int i,j,k,m,c,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++)
            ans[j++]=rep[k];
            c=m;
            i=0;
         }
      }
      else{
         ans[j]=str[c];
         j++;
         c++;
         m=c;
         i=0;
      }
   }
int main(){
   printf("\nEnter a Main String:\n");
   gets(str);
   printf("\nEnter a Patter String:\n");
   gets(pat);
   printf("\nEnter a Replacement String:\n");
   gets(rep);

   stringmatch();
```

```
    if(flag==1)
    printf("\nThe resultant string is :\n%s",ans);
    else
    printf("\nPattern String not found!!\n");

    return 0;
}
```

OUTPUT



```
Enter a Main String:
This is Data Structure lab

Enter a Patter String:
Data Structure

Enter a Replacement String:
Data structure with C

The resultant string is :
This is Data structure with C lab
```

PROGRAM 3

3.Develop 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 how Stack can be used to check Palindrome
d. Demonstrate Overflow and Underflow situations on Stack
e. Display the status of Stack
f. Exit
Support the program with appropriate functions for each of the above operations.

```
#include<stdio.h>
#include<stdlib.h>
#define max 4
int stack[max],i,itemdel;
int top = -1;
int flag=1;
int status = 0;

void push(int stack[] , int item){
  if(top==max-1){
    printf("\nStack is Full!");
  }
  else{
    stack[++top] = item ;
    status++;
  }
}

void pop(int stack[]){
  if(top==-1){
    printf("\nStack is Empty!");
  }
  else{
    itemdel = stack[top--];
    status--;
    printf("\nItem deleted = %d",itemdel);
  }
}

void display(int stack[]){
  if(top==-1){
    printf("\nStack is Empty!");
  }
  printf("\nStack contents are :\n");
  for(i=top;i>=0;i--){
    printf("|%d|\n",stack[i]);
  }
}

void pallindrome(int stack[]){
  display(stack);
  printf("\nReverse of stack contents are:\n");
  for(i=0;i<=top;i++){
    printf("|%d|\n",stack[i]);
  }

  for(i=0;i<=top/2;i++){
    if(stack[i]!=stack[top-i])
    flag=0;
    break;
  }
  if(flag==1)
  printf("\nStack is a Pallindrome.");
  else
  printf("\nStack is Not a Pallinrome.");
}

int main(){
  int ch,item;
  while(ch!=5){
    printf("\n1.Push\t2.Pop\t3.Display\t4.Pallindrome\t5.Exit\nEnter your choice:");
    scanf("%d",&ch);
    switch(ch){
      case 1:
      printf("\nEnter the element to be inserted:");
      scanf("%d",&item);
      push(stack,item);
      break;
      case 2:
      pop(stack);
      break;
      case 3:
      display(stack);
      break;
      case 4:
      pallindrome(stack);
      break;
      case 5:
      break;
      default :
      printf("\nInvalid Choice!!");
    }
  }
  return 0;
}
```

OUTPUT

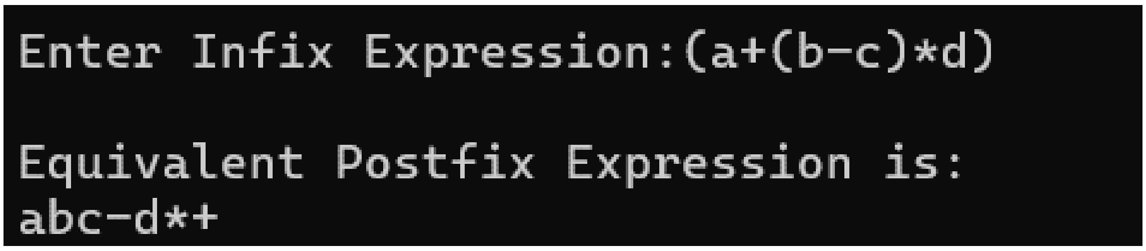
```
1.Push  2.Pop   3.Display       4.Pallindrome   5.Exit
Enter your choice:1

Enter the element to be inserted:1

1.Push  2.Pop   3.Display       4.Pallindrome   5.Exit
Enter your choice:1

Enter the element to be inserted:2

1.Push  2.Pop   3.Display       4.Pallindrome   5.Exit
Enter your choice:1

Enter the element to be inserted:2

1.Push  2.Pop   3.Display       4.Pallindrome   5.Exit
Enter your choice:1

Enter the element to be inserted:1

1.Push  2.Pop   3.Display       4.Pallindrome   5.Exit
Enter your choice:1

Enter the element to be inserted:5

Stack is Full!
1.Push  2.Pop   3.Display       4.Pallindrome   5.Exit
Enter your choice:3

Stack contents are :
|1|
|2|
|2|
|1|

1.Push  2.Pop   3.Display      4.Pallindrome   5.Exit
Enter your choice:4

Stack contents are :
|1|
|2|
|2|
|1|

Reverse of stack contents are:
|1|
|2|
|2|
|1|

Stack is a Pallindrome.
1.Push  2.Pop   3.Display      4.Pallindrome   5.Exit
Enter your choice:2

Item deleted = 1
1.Push  2.Pop   3.Display      4.Pallindrome   5.Exit
Enter your choice:3

Stack contents are :
|2|
|2|
|1|

1.Push  2.Pop   3.Display      4.Pallindrome   5.Exit
Enter your choice:2

Item deleted = 2
1.Push  2.Pop   3.Display      4.Pallindrome   5.Exit
Enter your choice:3

Stack contents are :
|2|
|1|

1.Push  2.Pop   3.Display      4.Pallindrome   5.Exit
Enter your choice:5
```

PROGRAM 4

4.Develop 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<stdlib.h>
#include<string.h>
#include<ctype.h>

int stkpre(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 inpre(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[]){
  char s[30],symbol;
  int i,j,top;
  top=-1;
  s[++top]='#';
  j=0;

  for(i=0;i<strlen(infix);i++){
    symbol = infix[i];
    while(stkpre(s[top])>inpre(symbol))
    postfix[j++]=s[top--];
    if(stkpre(s[top])!=inpre(symbol))
    s[++top]=symbol;
    else
    top--;
  }

  while(s[top]!='#'){
    postfix[j++] = s[top--];
  }
  postfix[j] = '\0';
}

int main(){
  char infix[30],postfix[30];
  printf("\nEnter Infix Expression:");
  scanf("%s",infix);
  printf("\nEquivalent Postfix Expression is:");
  infix_postfix(infix,postfix);
  printf("\n%s",postfix);
  return 0;
}
```

OUTPUT


```
Enter Infix Expression:(a+(b-c)*d)

Equivalent Postfix Expression is:
abc-d*+
```

PROGRAM 5A

5. Develop 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>
#include <ctype.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);
 case '%':
 return fmod(op1 , op2);
 default:
 return 0;
 }
}
int main() {
 double s[100];
 double res, op1, op2;
 int top = -1, i;
 char postfix[100], symbol;
 printf("\nEnter the postfix expression:\n");
 fgets(postfix, sizeof(postfix), stdin);
 for (i = 0; i < strlen(postfix); i++) {
 symbol = postfix[i];
 if (isspace(symbol))
 continue;
 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);
 return 0;
}
```

OUTPUT

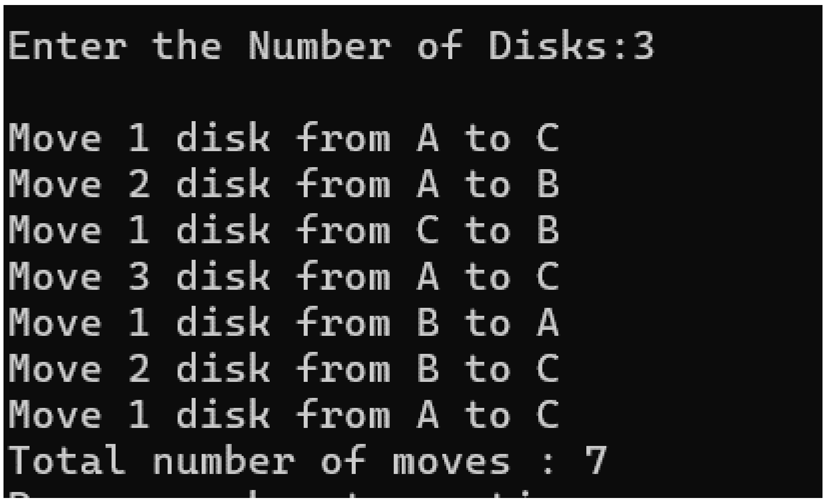
```
Enter the postfix expression:

2 3 1 * + 9 -


The result is : -4.000000
```

PROGRAM 5B

b. Solving Tower of Hanoi problem with n disks.

```
#include<stdlib.h>
#include<stdlib.h>
int count = 0;
void hanoi(int n,int src,int temp,int des){
  if(n==0)
  return;
  hanoi(n-1,src,des,temp);
  printf("\nMove %d disk from %c to %c",n,src,des);
  count++;
  hanoi(n-1,temp,src,des);
}

int main(){
  int n;
  printf("Enter the Number of Disks:");
  scanf("%d",&n);
  hanoi(n,'A','B','C');
  printf("\nTotal number of moves : %d",count);
  return 0;
}
```

OUTPUT



```
Enter the Number of Disks:3

Move 1 disk from A to C
Move 2 disk from A to B
Move 1 disk from C to B
Move 3 disk from A to C
Move 1 disk from B to A
Move 2 disk from B to C
Move 1 disk from A to C
Total number of moves : 7
```

PROGRAM 6

6. Develop 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
Support the program with appropriate functions for each of the above operations.

```
#include<stdio.h>
#include<stdlib.h>
#define max 4
int item,q[max],i;
int count=0;
int front=0;
int rear=-1;

void insert(int item,int *count,int *q,int*rear){
  if(*count==max){
    printf("\nCircular Queue is Full!");
  }
  else{
    *rear = (*rear+1)%max;
    q[*rear]=item;
    (*count)++;
  }
}

void del(int *count , int *q , int *front){
  if(*count == 0){
    printf("\nCircular Queue is Empty!");
  }
  else{
    int itemdel = q[*front];
    *front = (*front+1)%max;
    (*count)--;
    printf("\nItem Deleted : %d",itemdel);
  }
}

void display(int front , int count , int q[]){
  if(count==0){
    printf("\nCircular Queue is Empty!");
  }
  else{
    printf("\nContents of Queue are:\n");
    for(i=0;i<count;i++){
      printf("%d\t",q[front]);
      front = (front+1)%max;
    }
  }
}

int main(){
  int ch;
  while(1){
    printf("\n1.Insert\t2.Delete\t3.Display\t4.Exit\nEnter your choice:");
    scanf("%d",&ch);
    switch(ch){
      case 1:
      printf("\nEnter item to be inserted:");
      scanf("%d",&item);
      insert(item,&count,q,&rear);
      break;
      case 2:
      del(&count,q,&front);
      break;
      case 3:
      display(front,count,q);
      break;
      case 4:
      exit(0);
      default:printf("\nInvalid Choice!");
    }
  }
  return 0;
}
```

OUTPUT

```
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice:1

Enter item to be inserted:1

1.Insert        2.Delete        3.Display       4.Exit
Enter your choice:1

Enter item to be inserted:2

1.Insert        2.Delete        3.Display       4.Exit
Enter your choice:1
Enter item to be inserted:3

1.Insert	2.Delete	3.Display	4.Exit
Enter your choice:1

Enter item to be inserted:4

1.Insert	2.Delete	3.Display	4.Exit
Enter your choice:1

Enter item to be inserted:5

Circular Queue is Full!
1.Insert	2.Delete	3.Display	4.Exit
Enter your choice:3

Contents of Queue are:
1	2	3	4
1.Insert	2.Delete	3.Display	4.Exit
Enter your choice:2

Item Deleted : 1
1.Insert	2.Delete	3.Display	4.Exit
Enter your choice:2

Item Deleted : 2
1.Insert	2.Delete	3.Display	4.Exit
Enter your choice:3

Contents of Queue are:
3	4
1.Insert	2.Delete	3.Display	4.Exit
Enter your choice:4
```

PROGRAM 7

7. Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN,Name, Branch, Sem, PhNo
a. Create a SLL of N Students Data by using front insertion.
b. Display the status of SLL and count the number of nodes in it
c. Perform Insertion / Deletion at End of SLL
d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)
e. Exit

```
#include<stdio.h>
#include<stdlib.h>
#define max 5

struct node {
  char usn[11];
  char name[15];
  char branch[5];
  int sem;
  char phone[15];
  struct node *next;
};

typedef struct node NODE;

int countnodes(NODE *head){
  NODE *p;
  p=head;
  int count=0;
  while(p!=NULL){

    p=p->next;
    count++;
  }
  return count;
}

NODE* getnode(){
  NODE *newnode;
  newnode = (NODE*)malloc(sizeof(NODE));
  newnode->next=NULL;
  if(newnode==NULL){
    printf("Memory allocation failed!!");
    return 0;
  }
  else{
    printf("Enter USN:");
    scanf("%s",newnode->usn);
    printf("Enter Name:");
    scanf("%s",newnode->name);
    printf("Enter Branch:");
    scanf("%s",newnode->branch);
    printf("Enter Sem:");
    scanf("%d",&newnode->sem);
    printf("Enter Phone No:");
    scanf("%s",newnode->phone);
  }
  return newnode;
}


NODE* display(NODE* head){
  NODE *p;
  if(head==NULL){
    printf("No Student Details to Display !!\n");
    return head;
  }
  else{
    printf("\nUSN\t\tNAME\t\tBRANCH\t\tSEM\t\tPHONE NO.\n");
    p=head;
    while(p!=NULL){
      printf("%s\t%s\t\t%s\t\t%d\t\t%s\n",p->usn,p->name,p->branch,p->sem,p->phone);
      p=p->next;
    }
  }
  printf("\nThe number of nodes in the list is : %d",countnodes(head));
  return head;
}

NODE* insert_front(NODE *head){
  if(max == countnodes(head)){
    printf("Database Full !!\n");
    return head;
  }
  NODE *newnode;
  newnode = getnode();
  newnode->next = head;
  return newnode;
}

NODE* insert_rear(NODE *head){
  if(max == countnodes(head)){
    printf("Database Full !!\n");
    return head;
  }
  NODE *temp , *newnode;
  temp = head;
  newnode = getnode();
  if(head==NULL){
    return newnode;
  }
  else{
    while(temp->next!=NULL){
      temp=temp->next;
    }
    temp->next=newnode;
    return head;
  }
}

NODE* del_front(NODE *head){
  if(countnodes(head)==0){
    printf("Database Empty !!\n");
    return head;
  }
  NODE *temp;
  temp = head;
  head = head->next;
  free(temp);
  printf("Data Deleted!!\n");
  return head;
}

NODE* del_rear(NODE *head) {
    if (head == NULL) {
        printf("\nDatabase Empty !!\n");
        return head;
    }
    if (head->next == NULL) {
        free(head);
        printf("Data Deleted!!\n");
        return NULL;
    }
    NODE *temp = head;
    while (temp->next->next != NULL) {
        temp = temp->next;
    }
    free(temp->next);
    temp->next = NULL;
    printf("Data Deleted!!\n");
    return head;
}


NODE* del(NODE *head){
  int ch;
  while (ch!=3) {
    printf("1.del_front\t2.del_rear\t3.Exit\nEnter your choice:\t");
    scanf("%d",&ch);
    switch(ch){
      case 1 :head = del_front(head);
              head = display(head);
              break;
      case 2 :head = del_rear(head);
              head = display(head);
              break;
      case 3 :break;
    }
  }
  return head;
}

NODE* insert(NODE *head){
  int ch;
  while (ch!=3) {
    printf("1.insert_front\t2.insert_rear\t3.Exit\nEnter your choice:\t");
    scanf("%d",&ch);
    switch(ch){
      case 1 :head = insert_front(head);
              head = display(head);
              break;
      case 2 :head = insert_rear(head);
              head = display(head);
              break;
      case 3 :break;
    }
  }
  return head;
}

NODE* stack(NODE *head){
  int ch;
  do{
    printf("\nSSL used as Stack...");
    printf("\n 1.Insert at Front(PUSH) \t 2.Delete from Front(POP))\t3.Exit");
    printf("\nEnter your choice: ");
    scanf("%d", &ch);
    switch(ch){
      case 1: head=insert_front(head);
              head = display(head);
              break;
      case 2: head=del_front(head);
              head = display(head);
              break;
      case 3: break;
    }
  }while(ch!=3);
  return head;
}

NODE* create(NODE *head){
  NODE *newnode;
  if(head==NULL){
    newnode=getnode();
    head=newnode;
  }
  else{
    head = insert_front(head);
  }
  return  head;
}

int main(){
int ch, i, n;
NODE *head;
head=NULL;
printf("\n*----------StudentDatabase-----------*");
do{
  printf("\n 1.Create\t 2.Display\t 3.Insert\t 4.Delete\t 5.Stack\t 6.Exit");
  printf("\nEnter your choice: ");
  scanf("%d", &ch);
  switch(ch){
    case 1: printf("\nHow many student data you want to create: ");
            scanf("%d", &n);
            for(i=0;i<n;i++){
              printf("Enter Student%d Details:--\n",i+1);
              head = create(head);
            }
            break;
    case 2: head=display(head);
            break;
    case 3: head=insert(head);
            break;
    case 4: head=del(head);
            break;
    case 5: head=stack(head);
            break;
    case 6: break;
  }
}while(ch!=6);
return 0;
}
```

OUTPUT:

```
*----------StudentDatabase-----------*
 1.Create       2.Display      3.Insert       4.Delete       5.Stack        6.Exit
Enter your choice: 1

How many student data you want to create: 2

Enter Student1 Details:--
Enter USN:1BI22IS079
Enter Name:PRAKHAR
Enter Branch:ISE
Enter Sem:3
Enter Phone No:9406737945

Enter Student2 Details:--
Enter USN:1BI22IS0XX
Enter Name:XXXXXXX
Enter Branch:ISE
Enter Sem:3
Enter Phone No:9999999991

 1.Create       2.Display      3.Insert       4.Delete       5.Stack        6.Exit
Enter your choice: 2

USN             NAME           BRANCH         SEM            PHONE NO.
1BI22IS0XX      XXXXXXX        ISE            3              9999999991
1BI22IS079      PRAKHAR        ISE            3              9406737945

The number of nodes in the list is : 2
 1.Create       2.Display      3.Insert       4.Delete       5.Stack        6.Exit
Enter your choice: 3

1.insert_front  2.insert_rear   3.Exit
Enter your choice:      1
Enter USN:1BI22IS0FF
Enter Name:FFFFFFF
Enter Branch:ISE
Enter Sem:3
Enter Phone No:9191919191

USN             NAME           BRANCH         SEM            PHONE NO.
1BI22IS0FF      FFFFFFF        ISE            3              9191919191
1BI22IS0XX      XXXXXXX        ISE            3              9999999991
1BI22IS079      PRAKHAR        ISE            3              9406737945

The number of nodes in the list is : 3
1.insert_front  2.insert_rear   3.Exit
Enter your choice:      2
Enter USN:1BI22IS0RR
Enter Name:RRRRRRR
Enter Branch:ISE
Enter Sem:3
Enter Phone No:7894612301

USN             NAME          BRANCH        SEM           PHONE NO.
1BI22IS0FF      FFFFFFF       ISE           3             9191919191
1BI22IS0XX      XXXXXXX       ISE           3             9999999991
1BI22IS079      PRAKHAR       ISE           3             9406737945
1BI22IS0RR      RRRRRRR       ISE           3             7894612301

The number of nodes in the list is : 4
1.insert_front  2.insert_rear   3.Exit
Enter your choice:      3

 1.Create       2.Display       3.Insert       4.Delete       5.Stack       6.Exit
Enter your choice: 5

SSL used as Stack...
 1.Insert at Front(PUSH)        2.Delete from Front(POP))      3.Exit
Enter your choice: 1
Enter USN:1BI22IS0SS
Enter Name:SSSSSSS
Enter Branch:ISE
Enter Sem:3
Enter Phone No:1123456789

USN             NAME          BRANCH        SEM           PHONE NO.
1BI22IS0SS      SSSSSSS       ISE           3             1123456789
1BI22IS0FF      FFFFFFF       ISE           3             9191919191
1BI22IS0XX      XXXXXXX       ISE           3             9999999991
1BI22IS079      PRAKHAR       ISE           3             9406737945
1BI22IS0RR      RRRRRRR       ISE           3             7894612301

The number of nodes in the list is : 5

SSL used as Stack...
 1.Insert at Front(PUSH)        2.Delete from Front(POP))      3.Exit
Enter your choice: 2
Data Deleted!!

USN             NAME          BRANCH        SEM           PHONE NO.
1BI22IS0FF      FFFFFFF       ISE           3             9191919191
1BI22IS0XX      XXXXXXX       ISE           3             9999999991
1BI22IS079      PRAKHAR       ISE           3             9406737945
1BI22IS0RR      RRRRRRR       ISE           3             7894612301

The number of nodes in the list is : 4
SSL used as Stack...
```

```
 1.Insert at Front(PUSH)        2.Delete from Front(POP))     3.Exit
Enter your choice: 3

USN              NAME          BRANCH        SEM          PHONE NO.
1BI22IS0FF      FFFFFFF       ISE            3            9191919191
1BI22IS0XX    XXXXXXX     ISE            3            9999999991
1BI22IS079     PRAKHAR     ISE            3            9406737945
1BI22IS0RR     RRRRRRR     ISE            3            7894612301

The number of nodes in the list is : 4
 1.Create       2.Display      3.Insert       4.Delete       5.Stack        6.Exit
Enter your choice: 4

1.del_front    2.del_rear      3.Exit
Enter your choice:      1
Data Deleted!!

USN              NAME          BRANCH        SEM          PHONE NO.
1BI22IS0XX    XXXXXXX     ISE            3            9999999991
1BI22IS079     PRAKHAR     ISE            3            9406737945
1BI22IS0RR     RRRRRRR     ISE            3            7894612301

The number of nodes in the list is : 3
1.del_front    2.del_rear      3.Exit
Enter your choice:      2
Data Deleted!!

USN             NAME           BRANCH        SEM          PHONE NO.
1BI22IS0XX    XXXXXXX     ISE             3            9999999991
1BI22IS079     PRAKHAR     ISE             3            9406737945

The number of nodes in the list is : 2
1.del_front    2.del_rear      3.Exit
Enter your choice:      3

 1.Create       2.Display      3.Insert       4.Delete       5.Stack        6.Exit
Enter your choice: 2

USN             NAME          BRANCH        SEM          PHONE NO.
1BI22IS0XX    XXXXXXX     ISE             3            9999999991
1BI22IS079     PRAKHAR     ISE             3            9406737945

The number of nodes in the list is : 2
 1.Create       2.Display      3.Insert       4.Delete       5.Stack        6.Exit
```

PROGRAM 8

8. Develop a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: SSN,Name, Dept, Designation, Sal, PhNo
a. Create a DLL of N Employees Data by using end insertion.
b. Display the status of DLL and count the number of nodes in it
c. Perform Insertion and Deletion at End of DLL
d. Perform Insertion and Deletion at Front of DLL
e. Demonstrate how this DLL can be used as Double Ended Queue
f. Exit

```
#include<stdio.h>
#include<stdlib.h>
#define max 5
struct node{
  int ssn;
  char name[15];
  char dept[10];
  char desig[20];
  int salary;
  char phno[15];
  struct node *prev;
  struct node *next;
};
typedef struct node NODE;

int countnodes(NODE *head){
  NODE *temp;
  int count = 0;
  if(head==NULL){
    return 0;
  }
  else{

    temp=head;
    while(temp){
      count++;
      temp=temp->next;
    }
  }
  return count;
}
NODE* getnode(){
  NODE *newnode;
  newnode = (NODE*)malloc(sizeof(NODE));
  if(newnode==NULL){
    printf("\nMemory allocation failed!!\n");
  }
  else{
    newnode->prev=newnode->next=NULL;
    printf("Enter SSN:");
    scanf("%d",&newnode->ssn);
    printf("Enter Name:");
    scanf("%s",newnode->name);
    printf("Enter Department:");
    scanf("%s",newnode->dept);
    printf("Enter Designation:");
    scanf("%s",newnode->desig);
    printf("Enter Salary:");
    scanf("%d",&newnode->salary);
    printf("Enter PhoneNo.:");
    scanf("%s",newnode->phno);
  }
  return newnode;
}
NODE* display(NODE *head){
  if(countnodes(head)==0){
    printf("Database Empty!!");
    return head;
  }
  else{
    printf("SSN\tNAME\t\tDEPARTMENT\tDESIGNATION\tSALARY\tPHONENO\n");
    NODE *temp;
    temp = head;
    while(temp!=NULL){
      printf("%d\t%s\t\t%s\t\t%s\t%d\t%s\n",temp->ssn,temp->name,temp->dept,temp-
>desig,temp->salary,temp->phno);
      temp=temp->next;
    }
  }
  printf("Total number of nodes : %d\n",countnodes(head));
  return head;
}
NODE* insertrear(NODE *head){
  NODE *temp;
  NODE *newnode;
  newnode = getnode();
  if(head==NULL){
    return newnode;
  }
  else if (countnodes(head)==max){
    printf("Database Full!!");
    return head;
  }
  else{
    temp=head;
    while(temp->next!=NULL){
      temp=temp->next;
    }
    temp->next = newnode;
    newnode->prev = temp;
    return head;
  }
}
NODE* insertfront(NODE *head){
  NODE *newnode;
  newnode = getnode();
  if(head==NULL){
    return newnode;
  }
  else if (countnodes(head)==max){
    printf("Database Full!!");
    return head;
  }
  else{
    newnode->next=head;
    head->prev=newnode;
    return newnode;
  }
}
NODE* insert(NODE *head){
  int ch=0 ;
  while(ch!=3){
    printf("1insert_front\t2.insert_rear\t3.Exit\nEnter your choice:");
    scanf("%d",&ch);
    switch(ch){
      case 1:
      head = insertfront(head);
      head=display(head);
      break;
      case 2:
      head = insertrear(head);
      head=display(head);
      break;
      case 3:
      break;
    }
  }
  return head;
}
NODE* del_rear(NODE *head){
  if(head==NULL){
    printf("Database empty!\n");
    return head;
  }
  if(head->next==NULL){
    free(head->next);
    printf("Data Deleted!\n");
    return head;
  }
  NODE *temp;
  temp=head;
  while(temp->next->next!=NULL){
    temp=temp->next;
  }
  free(temp->next);
  temp->next=NULL;
  printf("Data Deleted!\n");
  return head;
}
NODE* del_front(NODE *head){
  if(head==NULL){
    printf("Database Empty\n");
    return head;
  }
  NODE *temp;
  temp=head;
  head=head->next;
  free(temp);
  printf("\nData Deleted!");
  return head;
}
NODE* del(NODE *head){
  int ch = 0; // Initialize ch
  while(ch!=3){
    printf("1.del_front\t2del_rear\t3.Exit\nEnter your choice:");
    scanf("%d",&ch);
    switch(ch){
      case 1:
      head = del_front(head);
      head=display(head);
      break;
      case 2:
      head =del_rear(head);
      head=display(head);
      break;
      case 3:
      break;
    }
  }
  return head;
}
NODE* dqueue(NODE *head) {
    int ch;
    while (1) {
        printf("\n DLL used as Double Ended Queue");
        printf("\n 1. Insert at Rear\n 2. Delete from Front\n 3. Insert at Front\n 4. Delete from
Rear\n 5. Display\n 6. Exit");
```

```
        printf("\nEnter your choice: ");
        scanf("%d", &ch);
        switch (ch) {
            case 1:
                head = insertrear(head);
                head=display(head);
                break;
            case 2:
                head = del_front(head);
                head=display(head);
                break;
            case 3:
                head = insertfront(head);
                head=display(head);
                break;
            case 4:
                head = del_rear(head);
                head=display(head);
                break;
            case 5:
                head = display(head);
                break;
            case 6:
                return head;
        }
    }
}

NODE* create(NODE *head){
  NODE *newnode;
  if(head==NULL){
    newnode=getnode(head);
    head=newnode;
  }
  else{
    head = insertfront(head);
  }
  return  head;
}
int main(){
  NODE *head;
  head = NULL;
  int ch ,n;
  while(1){
    printf("-------EMPLOYEE DATABASE SYSTEM ------");
    printf("\n1.Create\t2.Display\t3.Insert\t4.Delete\t5.Queue\t6.Exit\nEnter your choice:");
    scanf("%d",&ch);
    switch(ch){
      case 1:
      printf("How many records you want to create? : ");
```

```
        scanf("%d",&n);
        for(int i=0;i<n;i++){
          printf("\nEnter Employee%d Details:--\n",i+1);
          head = create(head);
        }
        break;
        case 2 :
        head = display(head);
        break;
        case 3:
        head = insert(head);
        break;
        case 4:
        head = del(head);
        break;
        case 5:
        head = dqueue(head);
        break;
        case 6:
        exit(0);
        default:
        printf("Invalid choice!!");
      }
    }
  return 0;
}
```

OUTPUT

```
-------EMPLOYEE DATABASE SYSTEM ------
1.Create      2.Display     3.Insert      4.Delete      5.Queue 6.Exit
Enter your choice:1
How many records you want to create? : 2
Enter Employee1 Details:--
Enter SSN:1
Enter Name:AAAAA
Enter Department:ISE
Enter Designation:INSTRUCTOR
Enter Salary:20000
Enter PhoneNo.:1234567890

Enter Employee2 Details:--
Enter SSN:2
Enter Name:BBBBB
Enter Department:ISE
Enter Designation:INSTRUCTOR
Enter Salary:25000
Enter PhoneNo.:4569832170
-------EMPLOYEE DATABASE SYSTEM ------
1.Create      2.Display     3.Insert      4.Delete      5.Queue 6.Exit
```

```
Enter your choice:2
SSN     NAME          DEPARTMENT     DESIGNATION    SALARY  PHONENO
2       BBBBB         ISE             INSTRUCTOR     25000      4569832170
1       AAAAA         ISE             INSTRUCTOR     20000      1234567890
Total number of nodes : 2
-------EMPLOYEE DATABASE SYSTEM ------
1.Create       2.Display      3.Insert       4.Delete       5.Queue 6.Exit
Enter your choice:3
1inset_front    2.insert_rear   3Exit
Enter your choice:1
Enter SSN:3
Enter Name:FFFFF
Enter Department:ISE
Enter Designation:INSTRUCTOR
Enter Salary:30000
Enter PhoneNo.:4569871230
SSN     NAME          DEPARTMENT     DESIGNATION    SALARY  PHONENO
3       FFFFF         ISE             INSTRUCTOR     30000      4569871230
2       BBBBB         ISE             INSTRUCTOR     25000      4569832170
1       AAAAA         ISE             INSTRUCTOR     20000      1234567890
Total number of nodes : 3
1inset_front    2.insert_rear   3Exit
Enter your choice:2
Enter SSN:4
Enter Name:RRRRR
Enter Department:ISE
Enter Designation:INSTRUCTOR
Enter Salary:35000
Enter PhoneNo.:4563271890
SSN     NAME          DEPARTMENT     DESIGNATION    SALARY  PHONENO
3       FFFFF         ISE             INSTRUCTOR     30000      4569871230
2       BBBBB         ISE             INSTRUCTOR     25000      4569832170
1       AAAAA         ISE             INSTRUCTOR     20000      1234567890
4       RRRRR         ISE             INSTRUCTOR     35000      4563271890
Total number of nodes : 4
1inset_front    2.insert_rear   3Exit
Enter your choice:3
-------EMPLOYEE DATABASE SYSTEM ------
1.Create       2.Display      3.Insert       4.Delete       5.Queue 6.Exit
Enter your choice:5

 DLL used as Double Ended Queue
 1. Insert at Rear
 2. Delete from Front
 3. Insert at Front
 4. Delete from Rear
 5. Display
 6. Exit
Enter your choice: 1
Enter SSN:5
Enter Name:INSERTF
Enter Department:ISE
Enter Designation:INSTRUCTOR
Enter Salary:40000
Enter PhoneNo.:1237894560
SSN     NAME           DEPARTMENT     DESIGNATION    SALARY  PHONENO
3       FFFFF          ISE            INSTRUCTOR     30000   4569871230
2       BBBBB          ISE            INSTRUCTOR     25000   4569832170
1       AAAAA          ISE            INSTRUCTOR     20000   1234567890
4       RRRRR          ISE            INSTRUCTOR     35000   4563271890
5       INSERTF        ISE            INSTRUCTOR     40000   1237894560
Total number of nodes : 5

 DLL used as Double Ended Queue
 1. Insert at Rear
 2. Delete from Front
 3. Insert at Front
 4. Delete from Rear
 5. Display
 6. Exit
Enter your choice: 2
Data Deleted!
SSN     NAME           DEPARTMENT     DESIGNATION    SALARY  PHONENO
2       BBBBB          ISE            INSTRUCTOR     25000   4569832170
1       AAAAA          ISE            INSTRUCTOR     20000   1234567890
4       RRRRR          ISE            INSTRUCTOR     35000   4563271890
5       INSERTF        ISE            INSTRUCTOR     40000   1237894560
Total number of nodes : 4
 DLL used as Double Ended Queue
 1. Insert at Rear
 2. Delete from Front
 3. Insert at Front
 4. Delete from Rear
 5. Display
 6. Exit
Enter your choice: 3
Enter SSN:6
Enter Name:INSERTQ
Enter Department:ISE
Enter Designation:INSTRUCTOR
Enter Salary:50000
Enter PhoneNo.:4536987120
SSN     NAME           DEPARTMENT     DESIGNATION    SALARY  PHONENO
6       INSERTQ        ISE            INSTRUCTOR     50000   4536987120
2       BBBBB          ISE            INSTRUCTOR     25000   4569832170
1       AAAAA          ISE            INSTRUCTOR     20000   1234567890
4       RRRRR          ISE            INSTRUCTOR     35000   4563271890
5       INSERTF        ISE            INSTRUCTOR     40000   1237894560
Total number of nodes : 5
 DLL used as Double Ended Queue
 1. Insert at Rear
 2. Delete from Front
 3. Insert at Front
 4. Delete from Rear
 5. Display
 6. Exit
Enter your choice: 4
Data Deleted!
SSN     NAME          DEPARTMENT     DESIGNATION     SALARY  PHONENO
6       INSERTQ        ISE           INSTRUCTOR      50000      4536987120
2       BBBBB          ISE           INSTRUCTOR      25000      4569832170
1       AAAAA          ISE           INSTRUCTOR      20000      1234567890
4       RRRRR          ISE           INSTRUCTOR      35000      4563271890
Total number of nodes : 4

 DLL used as Double Ended Queue
 1. Insert at Rear
 2. Delete from Front
 3. Insert at Front
 4. Delete from Rear
 5. Display
 6. Exit
Enter your choice: 5
SSN     NAME          DEPARTMENT     DESIGNATION     SALARY  PHONENO
6       INSERTQ        ISE           INSTRUCTOR      50000      4536987120
2       BBBBB          ISE           INSTRUCTOR      25000      4569832170
1       AAAAA          ISE           INSTRUCTOR      20000      1234567890
4       RRRRR          ISE           INSTRUCTOR      35000      4563271890
Total number of nodes : 4
 DLL used as Double Ended Queue
 1. Insert at Rear
 2. Delete from Front
 3. Insert at Front
 4. Delete from Rear
 5. Display
 6. Exit
Enter your choice: 6
-------EMPLOYEE DATABASE SYSTEM ------
1.Create       2.Display       3.Insert       4.Delete       5.Queue 6.Exit
Enter your choice:4
1del_front     2del_rear      3.Exit
Enter your choice:1
Data Deleted!
SSN     NAME          DEPARTMENT     DESIGNATION     SALARY  PHONENO
2       BBBBB          ISE           INSTRUCTOR      25000      4569832170
1       AAAAA          ISE           INSTRUCTOR      20000      1234567890
4       RRRRR          ISE           INSTRUCTOR      35000      4563271890
Total number of nodes : 3
1del_front     2del_rear      3.Exit
Enter your choice:2
```

```
Data Deleted!
SSN     NAME          DEPARTMENT     DESIGNATION    SALARY  PHONENO
2        BBBBB          ISE                  INSTRUCTOR     25000       4569832170
1        AAAAA          ISE                  INSTRUCTOR     20000       1234567890
Total number of nodes : 2
1del_front      2del_rear       3.Exit
Enter your choice:3
-------EMPLOYEE DATABASE SYSTEM ------
1.Create        2.Display       3.Insert        4.Delete        5.Queue 6.Exit
Enter your choice:6
```

PROGRAM 9

9. Develop a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes
a. Represent and Evaluate a Polynomial P(x,y,z) = 6x2y2z-4yz5+3x3yz+2xy5z- 2xyz3
b. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z)
Support the program with appropriate functions for each of the above operations.

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

struct node {
   int cf, px, py, pz;
   int flag;
   struct node *link;
};

typedef struct node NODE;

NODE* getnode() {
   NODE *x;
   x = (NODE*)malloc(sizeof(NODE));
   if(x == NULL) {
      printf("Insufficient memory\n");
      exit(0);
   }
   return x;
}

void display(NODE *head) {
   NODE *temp;
   if(head->link == head) {
      printf("Polynomial does not exist\n");
      return;
   }
   temp = head->link;
   printf("\n");
   while(temp != head) {
      printf("%d x^%d y^%d z^%d", temp->cf, temp->px, temp->py, temp->pz);
      if(temp->link != head)
         printf(" + ");
      temp = temp->link;
   }
   printf("\n");
}
```

```
NODE* insert_rear(int cf, int x, int y, int z, NODE *head) {
    NODE *temp, *cur;
    temp = getnode();
    temp->cf = cf;
    temp->px = x;
    temp->py = y;
    temp->pz = z;
    cur = head->link;
    while(cur->link != head) {
        cur = cur->link;
    }
    cur->link = temp;
    temp->link = head;
    return head;
}

NODE* read_poly(NODE *head) {
    int px, py, pz, cf, ch;
    printf("\nEnter coeff: ");
    scanf("%d", &cf);
    printf("\nEnter x, y, z powers(0-indicate NO term): ");
    scanf("%d%d%d", &px, &py, &pz);
    head = insert_rear(cf, px, py, pz, head);
    printf("\nIf you wish to continue press 1 otherwise 0: ");
    scanf("%d", &ch);
    while(ch != 0) {
        printf("\nEnter coeff: ");
        scanf("%d", &cf);
        printf("\nEnter x, y, z powers(0-indicate NO term): ");
        scanf("%d%d%d", &px, &py, &pz);
        head = insert_rear(cf, px, py, pz, head);
        printf("\nIf you wish to continue press 1 otherwise 0: ");
        scanf("%d", &ch);
    }
    return head;
}

NODE* add_poly(NODE *h1, NODE *h2, NODE *h3) {
    NODE *p1, *p2;
    int x1, x2, y1, y2, z1, z2, cf1, cf2, cf;
    p1 = h1->link;

    while(p1 != h1) {
        x1 = p1->px;
        y1 = p1->py;
        z1 = p1->pz;
        cf1 = p1->cf;
        p2 = h2->link;
        while(p2 != h2) {
            x2 = p2->px;
            y2 = p2->py;
            z2 = p2->pz;
            cf2 = p2->cf;
            if(x1 == x2 && y1 == y2 && z1 == z2) break;
            p2 = p2->link;
        }
        if(p2 != h2) {
            cf = cf1 + cf2;
            p2->flag = 1;
            if(cf != 0)
                h3 = insert_rear(cf, x1, y1, z1, h3);
        } else
            h3 = insert_rear(cf1, x1, y1, z1, h3);
        p1 = p1->link;
    }
    p2 = h2->link;
    while(p2 != h2) {
        if(p2->flag == 0)
            h3 = insert_rear(p2->cf, p2->px, p2->py, p2->pz, h3);
        p2 = p2->link;
    }
    return h3;
}

void evaluate(NODE *h) {
    NODE *head;
    int x, y, z;
    float result = 0.0;
    head = h->link;
    printf("\nEnter x, y, z, terms to evaluate:\n");
    scanf("%d%d%d", &x, &y, &z);
    while(head!= h) {
        result = result + (head->cf * pow(x, head->px) * pow(y, head->py) * pow(z, head->pz));
        head = head->link;
    }
    printf("\nPolynomial result is: %f", result);
}

int main() {
    NODE *h, *h1, *h2, *h3;
    int ch;
    while(1) {
        printf("\n\n1.Evaluate polynomial\n2.Add two polynomials\n3.Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &ch);
        switch(ch) {
            case 1:
                printf("\nEnter polynomial to evaluate:\n");
                h = getnode();
                h->link = h;
                h = read_poly(h);
                display(h);
                evaluate(h);
                break;
            case 2:
                printf("\nEnter the first polynomial:");
                h1 = getnode();
                h1->link = h1;
                h1 = read_poly(h1);
                printf("\nEnter the second polynomial:");
                h2 = getnode();
                h2->link = h2;
                h2 = read_poly(h2);
                h3 = getnode();
                h3->link = h3;
                h3 = add_poly(h1, h2, h3);
                printf("\nFirst polynomial is: ");
                display(h1);
                printf("\nSecond polynomial is: ");
                display(h2);
                printf("\nThe sum of 2 polynomials is: ");
                display(h3);
                break;
            case 3:
                exit(0);
                break;
            default:
                printf("\nInvalid entry");
                break;
        }
    }
    return 0;
}
```

OUTPUT

```
1.Evaluate polynomial
2.Add two polynomials
3.Exit
Enter your choice: 1

Enter polynomial to evaluate:

Enter coeff: 6

Enter x, y, z powers(0-indicate NO term): 2 2 1

If you wish to continue press 1 otherwise 0: 1

Enter coeff: -4
Enter x, y, z powers(0-indicate NO term): 0 1 5

If you wish to continue press 1 otherwise 0: 1

Enter coeff: 3

Enter x, y, z powers(0-indicate NO term): 3 1 1

If you wish to continue press 1 otherwise 0: 1

Enter coeff: 2

Enter x, y, z powers(0-indicate NO term): 1 5 1

If you wish to continue press 1 otherwise 0: 1

Enter coeff: -2

Enter x, y, z powers(0-indicate NO term): 1 1 3

If you wish to continue press 1 otherwise 0: 0

6 x^2 y^2 z^1 + -4 x^0 y^1 z^5 + 3 x^3 y^1 z^1 + 2 x^1 y^5 z^1 + -2 x^1 y^1 z^3

Enter x, y, z, terms to evaluate:
1 1 1

Polynomial result is: 5.000000

1.Evaluate polynomial
2.Add two polynomials
3.Exit
Enter your choice: 2

Enter the first polynomial:
Enter coeff: 4

Enter x, y, z powers(0-indicate NO term): 2 2 2

If you wish to continue press 1 otherwise 0: 1

Enter coeff: 3

Enter x, y, z powers(0-indicate NO term): 1 1 2

If you wish to continue press 1 otherwise 0: 0

Enter the second polynomial:
Enter coeff: 6
Enter x, y, z powers(0-indicate NO term): 2 2 2

If you wish to continue press 1 otherwise 0: 0

First polynomial is:
4 x^2 y^2 z^2 + 3 x^1 y^1 z^2

Second polynomial is:
6 x^2 y^2 z^2

The sum of 2 polynomials is:
10 x^2 y^2 z^2 + 3 x^1 y^1 z^2
1.Evaluate polynomial
2.Add two polynomials
3.Exit
Enter your choice: 3
```

PROGRAM 10

10. Develop 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, 5, 2
b. Traverse the BST in Inorder, Preorder and Post Order
c. Search the BST for a given element (KEY) and report the appropriate message
d. Exit

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

NODE* insert(int item , NODE *root){
  NODE *temp , *cur , *prev ;
  temp = (NODE*)malloc(sizeof(NODE));
  temp->data = item;
  temp->llink = temp->rlink = NULL;
  if(root == NULL){
    return temp;
  }
  prev = NULL;
  cur = root;
  while(cur != NULL){
    prev = cur ;
    if(cur->data >= item) cur = cur->llink;
    else cur = cur->rlink;
  }
  if(item < prev->data){
    prev->llink = temp;
  }
  else{
    prev->rlink = temp;
  }
  return root;
}

NODE* construct_bsf(NODE*root){
  int n , a;
  printf("Enter the number of element:");
  scanf("%d",&n);
  printf("Enter Elements : ");
  for(int i = 0 ; i<n ; i++ ){
    scanf("%d",&a);
    root = insert(a,root);
  }
  printf("Tree created Successfully !\n");
  return root ;
}
int search_item(NODE *root , int key){
  if(root == NULL) {
    printf("Tree Empty!");
    return root;
  }
  NODE *cur ;
  cur = root;
  while(cur){
    if(cur->data == key){
       printf("Key found!");
       break;
    }
    else{
      if(key < cur->data) cur = cur->llink;
      else cur=cur->rlink;
    }
  }
  if(cur == NULL) printf("KEY not found!");
}
void preorder(NODE *root){
  if(root != NULL){
    printf("%d\t",root->data);
    preorder(root->llink);
    preorder(root->rlink);
  }
}
void postorder(NODE *root){
  if(root != NULL){
    postorder(root->llink);
    postorder(root->rlink);
    printf("%d\t",root->data);
  }
}
void inorder(NODE *root){
  if(root != NULL){
    inorder(root->llink);
    printf("%d\t",root->data);
    inorder(root->rlink);
  }
}

void main(){
  NODE *root;
  root = NULL;
  int ch , key;
  while(1){
    printf("\n1.Construct Tree\n2.Preorder\n3.Postorder\n4.Inorder\n5.Search\n6.Exit\nEnter
your Choice:");
    scanf("%d",&ch);
    switch(ch){
      case 1:root = construct_bsf(root);
             break;
      case 2:preorder(root);
             break;
      case 3:postorder(root);
             break;
      case 4:inorder(root);
             break;
      case 5:printf("Enter key to search:");
             scanf("%d",&key);
             search_item(root,key);
             break;
      case 6:exit(0);
      default:printf("Invalid Choice");

    }
  }
}
```

OUTPUT

```
1.Construct Tree
2.Preorder
3.Postorder
4.Inorder
5.Search
6.Exit
Enter your Choice:1
Enter the number of element:4
Enter Elements : 1 2 3 4
Tree created Successfully !

1.Construct Tree
2.Preorder
3.Postorder
4.Inorder
5.Search
6.Exit
Enter your Choice:2
1       2       3       4

1.Construct Tree
2.Preorder
3.Postorder
4.Inorder
```

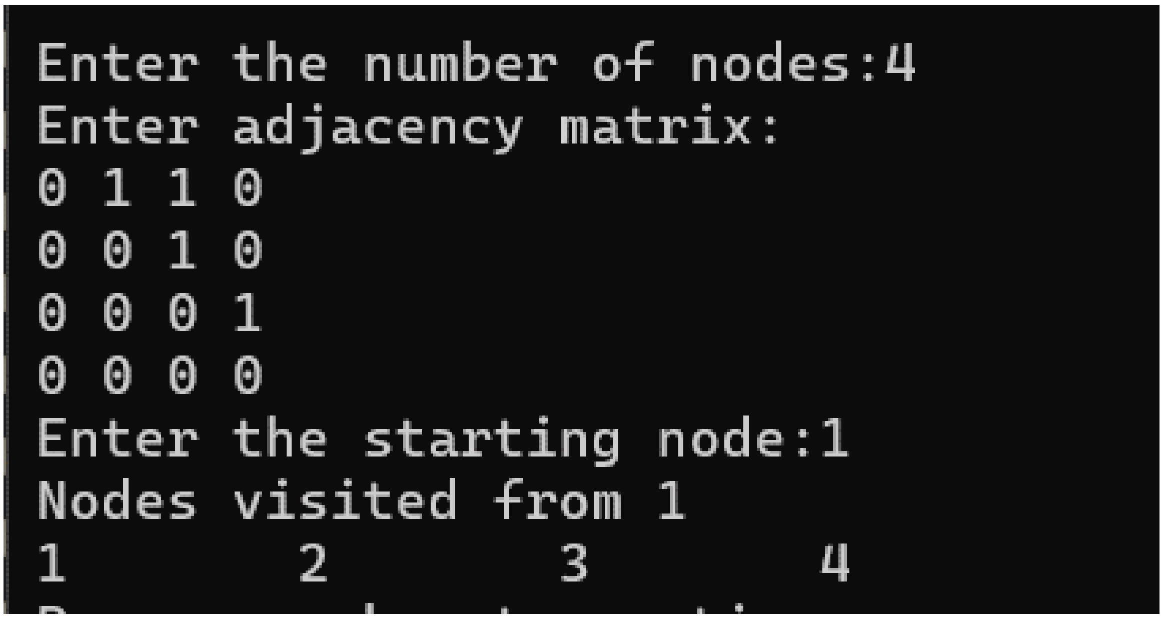
```
5.Search
6.Exit
Enter your Choice:3
4	3	2	1

1.Construct Tree
2.Preorder
3.Postorder
4.Inorder
5.Search
6.Exit
Enter your Choice:4
1	2	3	4

1.Construct Tree
2.Preorder
3.Postorder
4.Inorder
5.Search
6.Exit
Enter your Choice:5
Enter key to search:1
Key found!

1.Construct Tree
2.Preorder
3.Postorder
4.Inorder
5.Search
6.Exit
Enter your Choice:5
Enter key to search:6
KEY not found!

1.Construct Tree
2.Preorder
3.Postorder
4.Inorder
5.Search
6.Exit
Enter your Choice:6
```

PROGRAM 11A

11. Develop a Program in C for the following operations on Graph(G) of Cities
a. Create a Graph of N cities using Adjacency Matrix.

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

int f,r,a[10][10],q[20],v,u,n;
int s[10]={0};

void bfs(){
  r=-1,f=0;
  printf("Nodes visited from %d\n",u);
  s[u]=1;
  q[++r]=u;
  printf("%d\t",u);
  while(f<=r){
    u=q[f++];
    for(v=1;v<=n;v++){
      if(a[u][v]==1){
        if(s[v]==0){
          printf("%d\t",v);
          q[++r]=v;
          s[v]=1;
        }
      }
    }
  }

}
int main(){
  printf("Enter the number of nodes:");
  scanf("%d",&n);
  printf("Enter adjacency matrix:\n");
  for(int i=1;i<=n;i++){
    for(int j=1;j<=n;j++){
      scanf("%d",&a[i][j]);
    }
  }
  printf("Enter the starting node:");
  scanf("%d",&u);
  bfs();
  return 0;
}
```

OUTPUT



```
Enter the number of nodes:4
Enter adjacency matrix:
0 1 1 0
0 0 1 0
0 0 0 1
0 0 0 0
Enter the starting node:1
Nodes visited from 1
1       2       3       4
```

PROGRAM 11B

b. Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method.

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

int f,r,a[10][10],q[20],v,u,n;
int visited[10]={0};

void dfs(int u){
  visited[u]=1;
  for(v=1;v<=n;v++){
    if(visited[v]==0 && a[u][v]==1){
      printf("%d\t",v);
      dfs(v);
    }
  }
}
int main(){
  printf("Enter the number of nodes:");
  scanf("%d",&n);
  printf("Enter adjacency matrix:\n");
  for(int i=1;i<=n;i++){
    for(int j=1;j<=n;j++){
      scanf("%d",&a[i][j]);
    }
  }
  printf("Enter the starting node:");
  scanf("%d",&u);
  printf("Nodes visited from %d \n ",u);
  printf("%d\t",u);
  dfs(u);
  return 0;
}
```

OUTPUT

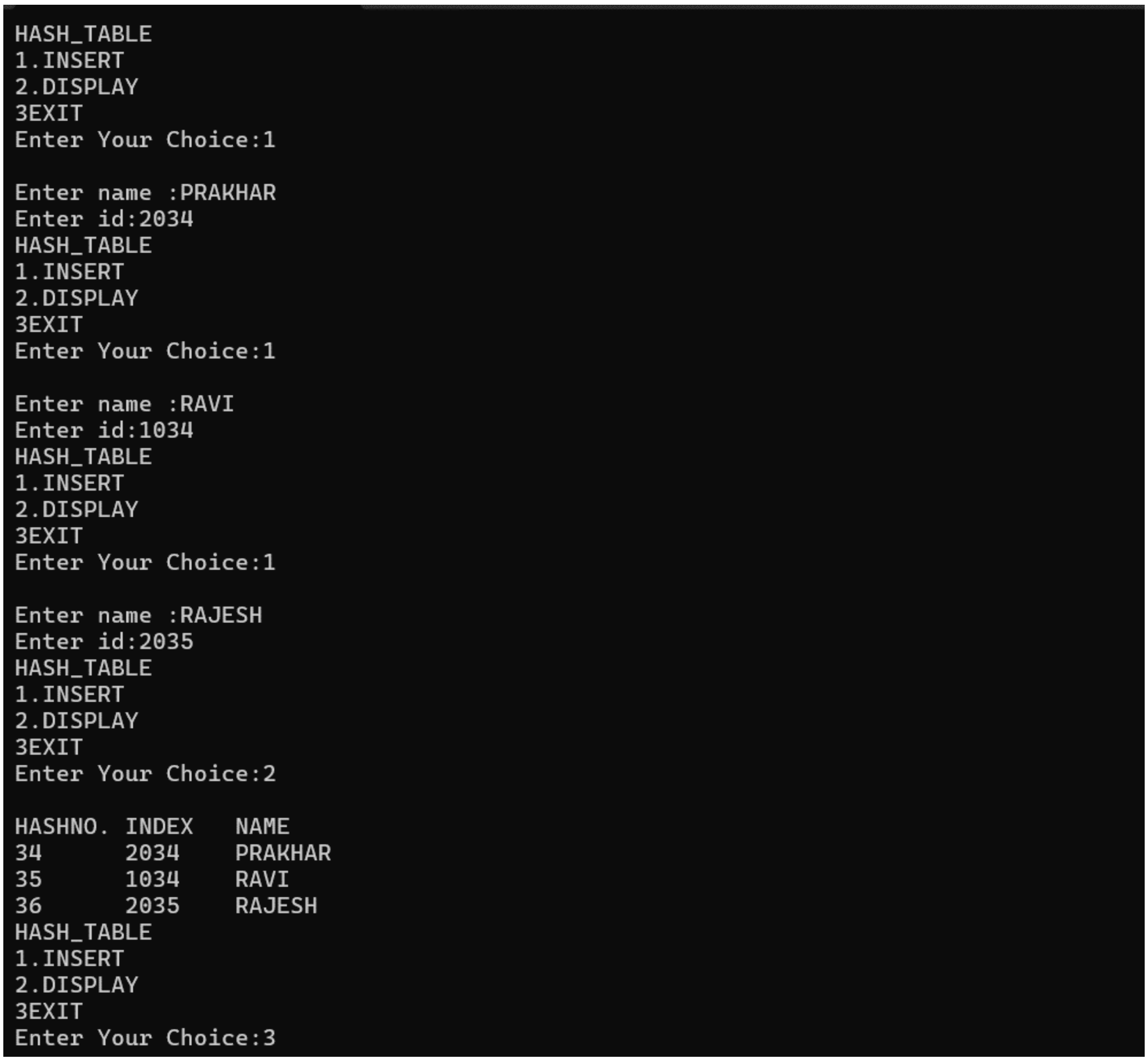
```
Enter the number of nodes:4
Enter adjacency matrix:
0 1 1 0
0 0 1 0
0 0 0 1
0 0 0 0
Enter the starting node:1
Nodes visited from 1
 1  2   3   4
```

PROGRAM 12

12. Given a File of N employee records with a set K of Keys(4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table(HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are Integers. Design and develop a Program in C that uses Hash function H: K →L as H(K)=K mod m (remainder method),and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing.

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

#define hashsize 100
struct employee{
  char name[30];
  int id;
};

typedef struct employee EMPLOYEE;

void create_hash_table(EMPLOYEE a[]){
  for(int i=0;i<hashsize;i++)
  a[i].id = 0;
}

int H(k){
  return k % hashsize;
}

void insert_table(EMPLOYEE a[] , int id , char name[]){
  int i , index , hashvalue ;
  hashvalue = H(id);
  for(i=0;i<hashsize;i++){
    index = (hashvalue+i)%hashsize;
    if(a[index].id == 0){
      a[index].id = id;
      strcpy(a[index].name,name);
      break;
    }
  }
  if(i==hashsize){
    printf("\nHASH TABLE FULL!!\n");
  }
}

void display(EMPLOYEE a[]){
  printf("\nHASHNO.\tINDEX\tNAME\n");
  for(int i= 0;i<hashsize;i++){
    if(a[i].id!=0){
      printf("%d\t%d\t%s\n",i,a[i].id,a[i].name);
    }
  }
}

int main(){
  EMPLOYEE a[hashsize];
  int ch , id ;
  char name[30];
  create_hash_table(a);
  while(1){
    printf("HASH_TABLE\n1.INSERT\n2.DISPLAY\n3EXIT\nEnter Your Choice:");
    scanf("%d",&ch);
    switch(ch){
      case 1:
      printf("\nEnter name :");
      scanf("%s",name);
      printf("Enter id:");
      scanf("%d",&id);
      insert_table(a,id,name);
      break;
      case 2:
      display(a);
      break;
      case 3:
      exit(0);

    }
  }
  return 0;
}
```

OUTPUT



```
HASH_TABLE
1.INSERT
2.DISPLAY
3EXIT
Enter Your Choice:1

Enter name :PRAKHAR
Enter id:2034
HASH_TABLE
1.INSERT
2.DISPLAY
3EXIT
Enter Your Choice:1

Enter name :RAVI
Enter id:1034
HASH_TABLE
1.INSERT
2.DISPLAY
3EXIT
Enter Your Choice:1

Enter name :RAJESH
Enter id:2035
HASH_TABLE
1.INSERT
2.DISPLAY
3EXIT
Enter Your Choice:2

HASHNO. INDEX   NAME
34      2034    PRAKHAR
35      1034    RAVI
36      2035    RAJESH
HASH_TABLE
1.INSERT
2.DISPLAY
3EXIT
Enter Your Choice:3
```