DSA LAB MANUAL

1.Develop a Programme in C for the following:

a) Declares 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 3 fields. The first field is the name of the day (a dynamically allocated string). The second field is the date of the day (an integer). The third field is the description of the activity for a particular day (a dynamically allocated string).

b) write function create() , read() and display() ; to create the calendar, to read the data from the keyboard and to print week's activity details report on screen.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// Define structure for a day
struct Day {
    char *name;       // Name of the day
    int date;         // Date of the day
    char *activity;   // Description of activity
};
int main() {
    int n = 7; // 7 days of the week
    // Dynamically allocate memory for 7 days
    struct Day *calendar = (struct Day*)malloc(n * sizeof(struct Day));
    if (calendar == NULL) {
        printf("Memory allocation failed!\n");
        return 1;
    }
```

```
    // Dynamically allocate memory for the name and activity strings for each day
    for (int i = 0; i < n; i++) {
        calendar[i].name = (char*)malloc(20 * sizeof(char));      // Allocate 20 chars for name
        calendar[i].activity = (char*)malloc(50 * sizeof(char));  // Allocate 50 chars for activity
    }
    printf("Calendar for 7 days is created successfully with dynamic memory.\n");
    // Example: Assign values (optional)
    strcpy(calendar[0].name, "Monday");
    calendar[0].date = 1;
    strcpy(calendar[0].activity, "Go to school");
    printf("Example Entry:\nDay: %s\nDate: %d\nActivity: %s\n",
           calendar[0].name, calendar[0].date, calendar[0].activity);
    // Free allocated memory
    for (int i = 0; i < n; i++) {
        free(calendar[i].name);
        free(calendar[i].activity);
    }
    free(calendar);
    return 0;
}
```

Output:

Calendar for 7 days is created successfully with dynamic memory.

Example Entry:

Day: Monday

Date: 1

Activity: Go to school

b.

```
#include <stdio.h>
// Structure for a day
struct Day {
    char name[20];      // Name of the day
    int date;           // Date of the day
    char activity[50];  // Activity description
};
// Function to read calendar data
void read(struct Day calendar[], int n) {
    for (int i = 0; i < n; i++) {
        printf("Enter name of day %d: ", i + 1);
        scanf(" %[^\n]", calendar[i].name);  // Read string with spaces
        printf("Enter date of %s: ", calendar[i].name);
        scanf("%d", &calendar[i].date);
        printf("Enter activity for %s: ", calendar[i].name);
        scanf(" %[^\n]", calendar[i].activity);
        printf("\n");
    }
}
// Function to display calendar data
void display(struct Day calendar[], int n) {
    printf("\n--- Weekly Activity Report ---\n");
    for (int i = 0; i < n; i++) {
        printf("Day: %s\n", calendar[i].name);
        printf("Date: %d\n", calendar[i].date);
```

```
        printf("Activity: %s\n\n", calendar[i].activity);
    }
}
int main() {
    int n = 7; // 7 days in a week
    struct Day calendar[7]; // Fixed-size array of 7 days
    // Read and display calendar
    read(calendar, n);
    display(calendar, n);
    return 0;
}
```

Output:

```
Enter name of day 1: Monday
Enter date of Monday: 1
Enter activity for Monday: reading

Enter name of day 2: Tuesday
Enter date of Tuesday: 2
Enter activity for Tuesday: exercise

Enter name of day 3: Wednesday
Enter date of Wednesday: 3
Enter activity for Wednesday: playing

Enter name of day 4: Thursday
Enter date of Thursday: 4
```

Enter activity for Thursday: studying

Enter name of day 5: Friday

Enter date of Friday: 5

Enter activity for Friday: singing

Enter name of day 6: Saturday

Enter date of Saturday: 6

Enter activity for Saturday: drawing

Enter name of day 7: Sunday

Enter date of Sunday: 5

Enter activity for Sunday: shopping

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 STK. 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

Code:

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

char str[100], pat[25], rep[25] ,res[25];
int i=0, j=0, k=0, l=0, ls=0, lp=0, lr=0, exist=0;
```

```
void read( );
void match( );


void main( )
{
  read( );
  match( );
}


void read( )
{
   printf("\n Enter main string\n");
   scanf("%s",str);

   printf("\n Enter pattern string\n");
   scanf("%s",pat);

   printf("\n Enter Replace string\n");
   scanf("%s",rep);
}


void match( )
{
  while(str[ls]!='\0')
    ls++;
  while(pat[lp]!='\0')
    lp++;
  while(rep[lr]!='\0')
    lr++;
while(i<=ls-lp)
  {
    j=0;
    while(j<lp && pat[j]==str[i+j])
      j++;

      if(j==lp)
        {
           exist=1;
```

```
            while(l<i)
             res[k++]=str[l++];
            l=0;
            while(l<lr)
            res[k++]=rep[l++];

            i=i+lp;
            l=i;
          }
        else
         i++;
  }
     if(exist==0)
      {
       printf("\n Pattern does not exist \n");
       exit(0);
      }

   while(l<ls)
   res[k++]=str[l++];
   res[k]='\0';
   printf("\n The final string after replacing is %s \n",res);
}
```

OUTPUT 1:

Enter main string
STUDENT

Enter pattern string
STUDENT

Enter Replace string
DAY

The final string after replacing is DAY

OUTPUT 2:

Enter main string
STUDENT

Enter pattern string
DAY

Enter Replace string
DATE

Pattern does not exist

3. Develop a menu driven Program in C for the following operations on STACK of Integers

(Array Implementation of Stack with maximum size MAXI
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

Code:

```
#include<stdio.h>
#include<stdlib.h>
#define max 20

int s[max],top=-1,ch,i,temp,elem;

void push( );
void pop( );
void palindrome( );
void display( );

void main( )
{
 while(1)
        {
          printf("\n \n Stack operations\n 1:Push\n 2:pop\n 3:Palindrome\n 4:display\n 5:exit\n");
```

```
            printf("Enter choice \n");
           scanf("%d",&ch);
            switch(ch)
               {
                    case 1:push( );
                        break;
                    case 2:pop( );
                     break;
                    case 3:palindrome( );
                              break;
                   case 4:display( );
                    break;
                   case 5:exit(0);
               }
          }
}
```

```
void push( )
{
 printf("Enter the element to be inserted\n");
 scanf("%d",&elem);
 if(top==max-1)
   printf("Stack overflow\n");
 else
   s[++top]=elem;
}

void pop( )
{
 if(top==-1)
   printf("Stack underflow\n");
 else
   top--;
}
void palindrome( )
{
    printf("Enter a number to check palindrome \n");
```

```
   scanf("%d",&elem);
   top=-1;
   temp=elem;
    while(temp)
       {
        s[++top]=temp%10;
       temp=temp/10;
       }
 temp=elem;
 while(temp)
     {
       if(s[top--]!=(temp%10))
        {
          printf("It is not palindrome\n");
          return;
        }
      temp=temp/10;
    }
   printf("It is palindrome\n");
}

void display( )
{
  printf("stack elements are \n");
  for(i=0;i<=top;i++)
  printf("%d\t",s[i]);
}
```

output

```
Stack operations
 1:Push
 2:pop
 3:Palindrome
 4:display
 5:exit
Enter choice
1
Enter the element to be inserted
10
```

```
 Stack operations
 1:Push
 2:pop
 3:Palindrome
 4:display
 5:exit
Enter choice
1
Enter the element to be inserted
20

Stack operations
 1:Push
 2:pop
 3:Palindrome
 4:display
 5:exit
Enter choice
4
stack elements are
10        20

 Stack operations
 1:Push
 2:pop
 3:Palindrome
 4:display
 5:exit
Enter choice
2


Stack operations
 1:Push
 2:pop
 3:Palindrome
 4:display
 5:exit
Enter choice
4
```

```
stack elements are
10

 Stack operations
 1:Push
 2:pop
 3:Palindrome
 4:display
 5:exit
Enter choice
2

Stack operations
 1:Push
 2:pop
 3:Palindrome
 4:display
 5:exit
Enter choice
2
Stack underflow

 Stack operations
 1:Push
 2:pop
 3:Palindrome
 4:display
 5:exit
Enter choice
3
Enter a number to check palindrome
121
It is palindrome
```

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

Code:

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

```
#include<string.h>

int F(char  ch)
{
   switch(ch)
    {
          case  '+':
          case  '-':return 2;
          case  '*':
        case  '/':
        case  '%':return 4;
        case  '$':
        case  '^':return 5;
        case  '(':return 0;
        case  '#':return -1;
        default: return 8;
    }
}
int G(char  ch)
{
      switch(ch)
   {
     case  '+':
     case  '-':return 1;
     case  '*':
     case  '/':
     case  '%':return 3;
     case  '$':
     case  '^':return 6;
     case  '(':return 9;
     case  ')':return 0;
     default:  return 7;
   }
}


void main( )
   {
      char s[20], Infix[100], postfix[100];
       int top=-1, j=0, i;
       printf("\nEnter the infix statement:\n");
```

```
    scanf("%s",infix);
    s[++top]='#';

   for(i=0;i<strlen(infix);i++)
    {
      while(F(s[top])>G(infix[i]))
       postfix[j++]=s[top--];

       if (F(s[top])!=G(infix[i]))
        s[++top]=infix[i];
      else
       top--;
     }
    while(s[top]!='#')
     postfix[j++]=s[top--];

     postfix[j]='\0';
     printf("\nThe postfix expession is %s \n",postfix);
}
```

Output:

Enter the infix statement:

A+(B*C-(D/E^F)*G)*H

The postfix expression is

ABC*DEF^/G*-H*+

5. Design, Develop and Implement a Program in C for the following Stack Applications

a. Evaluation of Suffix exp with single digit op & operators: +, -, *, /, %, ^

b. Solving Tower of Hanoi problem with n disks.

Code:

```
#include<stdio.h>

#include<string.h>

#include<math.h>
```

```
void main( )
{
   char postfix[20];
   int result,i,s[20],op1,op2,top=-1;

   printf("Enter postfix expression\n");
   scanf("%s",postfix);
   for(i=0;i<strlen(postfix);i++)
      {
         if(isdigit(postfix[i]))
            s[++top]=postfix[i]-'0';
        else
             {
          op2=s[top--];
               op1=s[top--];
               switch(postfix[i])
                    {
                     case '+':result=op1+op2;
                              break;
                     case '-':result=op1-op2;
                             break;
                     case '*':result=op1*op2;
                              break;
                      case '/':result=op1/op2;
                              break;
                     case '%':result=op1%op2;
```

```
                    break;
                case '^':result=pow(op1,op2);
                }
              s[++top]=result;
            }
   }
 printf("result =%d",result);
}
```

output:

Enter postfix expression

231*+9-

result =-4

b.

```
#include<stdio.h>
tower(int  n,char  s,char  t,char  d)
  {
       if(n==0) return;
          tower(n-1,s,d,t);
          printf("\n Move Disc %d from %c to %c",n,s,d);
          tower(n-1,t,s,d);
   }

void main( )
  {
```

```
    int n;
    printf("Enter no of discs\n");
    scanf("%d",&n);
    tower(n,'A','B','C');
}
```

OUTPUT:

Enter no of discs

3

Move Disc 1 from A to C

Move Disc 2 from A to B

Move Disc 1 from C to B

Move Disc 3 from A to C

Move Disc 1 from B to A

Move Disc 2 from B to C

Move Disc 1 from A to C

6. Design, Develop and Implement a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)

a. Insert an Element on to Circular QUEUE

b. Delete an Element from Circular QUEUE

c. Demonstrate Overflow and Underflow situations on Circular QUEUE

d. Display the status of Circular QUEUE

e. Exit

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

Code:

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

void  insert( );
 void delete( );
 void display( );

 char q[MAX],elem;
 int c=0,f=0,r=-1,i,front,ch;

 void main( )
{
    while(1)
       {
     printf("\n Circular queue operations \n 1:insert \n 2:detete\n3:display\n  4:exit \n");
     printf("enter choice\n");
     scanf("%d",&ch);
     switch(ch)
            {
      case 1:insert( );
               break;
      case 2:delete( );
                break;
      case 3:display( );
```

```
                break;
       case 4:exit(0);
             }
        }
}
void  insert( )
{
   if(c==MAX)
      {
        printf("Queue is full!");
           return;
        }
    printf("Enter element to be insert:\t");
    scanf(" %c",&elem);
    r=(r+1)%MAX;
    q[r]=elem;
    c++;
}
void delete( )
{
   if(c==0)
      {
     printf("Queue is Empty!\n");
     return;
      }
    printf("Deleted Element is: %c \n",q[f]);
```

```
    f=(f+1)%MAX;
    c--;
}
void display( )
{
   if(c==0)
    {
      printf("Queue is Empty!\n");
      return;
    }
    printf("Queue elements are:\n");
    front=f;
    for(i=0;i<c;i++)
     {
        printf("%c\t",q[front]);
        front=(front+1)%MAX;
      }
    printf("\n");
}
```

Output:

Circular queue operations

1:insert

2:detete

3:display

4:exit

enter choice

1

Enter element to be insert: a

Circular queue operations

1:insert

2:detete

3:display

4:exit

enter choice

1

Enter element to be insert: b

Circular queue operations

1:insert

2:detete

3:display

4:exit

enter choice

1

Enter element to be insert: c

Circular queue operations

1:insert

2:detete

3:display

4:exit

enter choice

1

Enter element to be insert: d

Circular queue operations

1:insert

2:detete

3:display

4:exit

enter choice

1

Enter element to be insert: e

Circular queue operations

1:insert

2:detete

3:display

4:exit

enter choice

1

Queue is full!

Circular queue operations

1:insert

2:detete

3:display

4:exit

enter choice

3

Queue elements are:

a b c d e

Circular queue operations

1:insert

2:detete

3:display

4:exit

enter choice

2

Deleted Element is: a

Circular queue operations

1:insert

2:detete

3:display

4:exit

enter choice

2

Deleted Element is: b

Circular queue operations

1:insert

2:detete

3:display

4:exit

enter choice

2

Deleted Element is: c

Circular queue operations

1:insert

2:detete

3:display

4:exit

enter choice

2

Deleted Element is: d

Circular queue operations

1:insert

2:detete

3:display

4:exit

enter choice

2

Deleted Element is: e

Circular queue operations

1:insert

2:detete

3:display

4:exit

enter choice

2

Queue is Empty!

Circular queue operations

1:insert

2:detete

3:display

4:exit

enter choice

4

7. Design, Develop and Implement 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

code:

```
#include<stdio.h>
#include<stdlib.h>
struct student
{
 char name[15],sem[15],usn[15],branch[15],phone[15];
```

```
  struct student *link;
};
typedef struct student *NODE;

main( )
{
 NODE first=NULL,temp,cur,prev;
 int choice,c;

   while(1)
       {
    printf("\n Operations of Singly linked list \n");
    printf(" \n 1:insert front\n 2:insert rear \n 3:delete front\n
                4:delete rear \n 5:display\n 6:exit \n");
    printf("\n Enter your choice \n");
    scanf("%d",&choice);
       switch(choice)
              {
                  case 1:temp=(NODE)malloc(sizeof(struct student));
printf("\n Enter name,sem,usn,branch and phone no \n");
scanf("%s %s %s %s %s",&temp->name,&temp->sem,
         &temp->usn,&temp->branch,&temp->phone);

                        temp->link=first;
                        first=temp;
                        break;
```

```
            case 2:temp=(NODE)malloc(sizeof(struct student));
      printf("\n Enter name,sem,usn,branch and phone no \n");
      scanf("%s %s %s %s %s",&temp->name,&temp->sem,
              &temp->usn,&temp->branch,&temp->phone);
                  temp->link=NULL;
                   if(first==NULL)
        {
         first=temp;
         break;
        }
         cur=first;
         while(cur->link!=NULL)
         cur=cur->link;

         cur->link=temp;
         break;
   case 3:if(first==NULL)
        {
         printf("\n List is empty \n");
         break;
        }
        temp=first;
        printf("\n First student details in the list is deleted\n");
        first=first->link;
        free(temp);
```

```
        break;


case 4:if(first==NULL)
        {
         printf("\n List is empty \n");
         break;
        }
        if(first->link==NULL)
        {
         printf(" List contains one student details,that is deleted");
         first=NULL;
         break;
        }

        prev=NULL;
        cur=first;
        while(cur->link!=NULL)
         {
           prev=cur;
           cur=cur->link;
         }
        prev->link=NULL;
        printf("\n Last student details in the list is deleted \n");
        free(cur);
        break;
```

```
                case 5:c=0;
            if(first==NULL)
            {
             printf("\n list is empty \n");
             break;
            }

            printf("\n The contents of linked list are \n");
            printf("name \t\tsem\t\t\tusn\t\t branch\t\t phone no ");
            printf(".....\n");
            temp=first;
            while(temp!=NULL)
            {
             printf("%-20s %-20s %-20s %-20s %-20s", temp->name, temp->sem,temp->usn, ->branch,temp->phone);
           printf("\n");
             temp=temp->link;
             c++;
            }
            printf("\n");
            printf("\n number of students= %d \n",c);
            break;
                    case 6:exit(0);

                  }
        }
```

```
}
```

Output:

```
Operations of Singly linked list

 1:insert front
 2:insert rear
 3:delete front
 4:delete rear
5:display
6:exit

 Enter your choice
1

 Enter name,sem,usn,branch and phone no
manoj
3rd
4mk24cs046
cse
1235468711

 Operations of Singly linked list

 1:insert front
 2:insert rear
```

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

2

Enter name,sem,usn,branch and phone no

akhil

3rd

4mk24cs00

cse

8054759545

Operations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

3

First student details in the list is deleted

Operations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

4

List contains one student details,that is deleted

Operations of Singly linked list

1:insert front

2:insert rear

3:delete frontOperations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

1

Enter name,sem,usn,branch and phone no

manoj

3rd

4mk24cs046

cse

1235468711

Operations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

2

Enter name,sem,usn,branch and phone no

akhil

3rd

4mk24cs00

cse

8054759545

Operations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

3

First student details in the list is deleted

Operations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

4

List contains one student details,that is deleted

Operations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

5

list is empty

Operations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

6

4:delete rear

5:display

6:exit

Enter your choice

5

list is empty

Operations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

output:

Operations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

1

Enter name,sem,usn,branch and phone no

manoj

3rd

4mk24cs046

cse

1235468711

Operations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

2

Enter name,sem,usn,branch and phone no

akhil

3rd

4mk24cs00

cse

8054759545

Operations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

3

First student details in the list is deleted

Operations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

4

List contains one student details,that is deleted

Operations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

5

list is empty

Operations of Singly linked list

1:insert front

2:insert rear

3:delete front

4:delete rear

5:display

6:exit

Enter your choice

6

8. Design, Develop and Implement 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

Code:

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

struct employee
{
  char name[15],ssn[15],dept[15],des[15],sal[15],phone[15];
  struct employee *llink,*rlink;
};
typedef struct employee *NODE;

main( )
{
 NODE first=NULL,temp,cur,prev;
 int choice,c;
```

```
while(1)
  {
      printf("\n Operations of Doubly linked list \n");
      printf(" \n 1:insert front\n 2:insert rear \n 3:delete front\n 4:delete rear \n
              5:display\n 6:exit \n");
      printf("\n Enter your choice \n");
      scanf("%d",&choice);
        switch(choice)
        {
            case 1:temp=(NODE)malloc(sizeof(struct employee));
                   printf("\n Enter name,ssn,dept,designation,salary and
                           phone no \n");
                   scanf("%s%s%s%s%s%s",&temp->name,&temp->ssn,
                          &temp->dept,&temp->des,&temp->sal,
                          &temp->phone);
                   temp->llink=temp->rlink=NULL;
                  if(first==NULL)
                     {
                      first=temp;
                      break;
                     }
                  temp->rlink=first;
                  first->llink=temp;
                  first=temp;
                  break;

             case 2:temp=(NODE)malloc(sizeof(struct employee));
                    printf("\n Enter name,ssn,dept,designation,salary and
                            phone no \n");
                    scanf("%s %s %s %s %s %s",&temp->name,&temp->ssn,
                           &temp->dept,&temp->des,&temp->sal,
                           &temp->phone);
                    temp->llink=temp->rlink=NULL;
                    if(first==NULL)
                    {
                     first=temp;
                     break;
                    }
                     cur=first;
                     while(cur->rlink!=NULL)
                     cur=cur->rlink;
```

```
            cur->rlink=temp;
            temp->llink=cur;
            break;

    case 3:if(first==NULL)
           {
            printf("\n List is empty \n");
            break;
           }
           temp=first;
           printf("\n First node in the list is deleted\n");
           first=first->rlink;
           first->llink=NULL;
           free(temp);
           break;

  case 4:if(first==NULL)
         {
          printf("\n List is empty \n");
          break;
         }
         if(first->rlink==NULL)
         {
          printf("\n List contains only one node, that is deleted\n");
          first=NULL;
          break;
         }




       prev=NULL;
        cur=first;

    while(cur->rlink!=NULL)
        {
          prev=cur;
          cur=cur->rlink;
        }
```

```
                    prev->rlink=NULL;
                    printf("\n Last node in the list is deleted\n");
                    free(cur);
                    break;

                case 5:c=0;
                    if(first==NULL)
                    {
                     printf("\n list is empty \n");
                     break;
                    }

                    printf("\n The contents of linked list are \n");
                    printf("Name    \t\tSSN\t\t\tDept\t\tDesignation\t\t
                            Sal\t\t   phone no \n");
                    printf(".....\n");
                    temp=first;
                    while(temp!=NULL)
                    {
                     printf("%-20s %-20s %-20s %-20s %-20s %-20s",
                     temp->name, temp->ssn,temp->dept,temp->des,
                     temp->sal,temp->phone);
                     printf("\n");
                     temp=temp->rlink;
                     c++;
                    }
                    printf("\n");
                    printf("\n Number of Employeess= %d \n",c);
                    break;
            case 6:exit(0);
        }
    }
}
```

OUTPUT:
Operations of Doubly linked list

1:insert front
2:insert rear
3:delete front

```
 4:delete rear
 5:display
 6:exit

 Enter your choice
1

 Enter name,ssn,dept,designation,salary and phone no
Navya
101
CSE
Professor
80000
1589632470

 Operations of Doubly linked list

 1:insert front
 2:insert rear
 3:delete front
 4:delete rear
 5:display
 6:exit

 Enter your choice
5

 The contents of linked list are
Name                    SSN                     Dept            Designation     Sal             phone no
.....
Navya           101             CSE             Professor       80000           1589632470


 Number of Employeess= 1

 Operations of Doubly linked list

 1:insert front
 2:insert rear
 3:delete front
 4:delete rear
```

```
 5:display
 6:exit

 Enter your choice
2

 Enter name,ssn,dept,designation,salary and phone no
Sahana
102
ECE
Assistant_Professor
70000

 Operations of Doubly linked list

 1:insert front
 2:insert rear
 3:delete front
 4:delete rear
 5:display
 6:exit

 Enter your choice
5

 The contents of linked list are
Name                    SSN                    Dept          Designation    Sal             phone no
.....................................................................
Navya          101          CSE          Professor        80000          1589632470
Sahana         102          ECE          Assistant_Professor        70000       5234562531


 Number of Employeess= 2

 Operations of Doubly linked list

 1:insert front
 2:insert rear
 3:delete front
 4:delete rear
 5:display
```

6:exit

Enter your choice
3

First node in the list is deleted

Operations of Doubly linked list

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:exit

Enter your choice
5

The contents of linked list are

Name	SSN	Dept	Designation	Sal	phone no
Sahana	102	ECE	Assistant_Professor	70000	5234562531

Number of Employeess= 1

Operations of Doubly linked list

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:exit

Enter your choice
4

List contains only one node, that is deleted

Operations of Doubly linked list

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:exit

Enter your choice
5

list is empty

Operations of Doubly linked list

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:exit

Enter your choice
6

9. Design, Develop and Implement a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes

a. **Represent and Evaluate a Polynomial**
   $P(x,y,z) = 6x^2y^2z-4yz^5+3x^3yz+2xy^5z-2xyz^3$
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.

Code:

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

```
struct poly
{
  int cf,px,py,pz;
  struct poly *link;
};
typedef struct poly *NODE;

void read(NODE  head);
void display(NODE  head);
void eval( );
void polysum( );

void main( )
{
    int sum=0,ch;
    while(1)
        {
  printf("\n 1:Polynomial Evaluation\n 2:Sum of two polynomial\n   3:exit\n");

  printf("\n Enter choice \n");
  scanf("%d",&ch);
  switch(ch)
              {
     case 1:eval( );
              break;
     case 2:polysum( );
               break;
     case 3:exit(0);
               }
        }
}
```

```
void read(NODE  head)
{
 NODE temp,cur;
 int n,i,x,y,z;
 printf("\nEnter no of terms \n");
 scanf("%d",&n);

            for(i=0;i<n;i++)
            {
             temp=(NODE)malloc(sizeof(struct poly));
             printf("\n Enter cf,px,py and pz of %i term\n",i+1);
             scanf("%d%d%d%d",&temp->cf,&temp->px,&temp->py,
                     &temp->pz);
             temp->link=head;

              cur=head->link;
              while(cur->link!=head)
              cur=cur->link;

              cur->link=temp;
            }
}

void display(NODE  head)
{
 NODE temp;
 if(head->link==head)
             {
              printf("\nPolynomial doesn't exist\n");
              return;
             }

            temp=head->link;
            while(temp!=head)
            {
             if(temp->cf<0)
                printf(" %d",temp->cf);
```

```
            else
                printf(" + %d",temp->cf);

            printf("x^%dy^%dz^%d",temp->px,temp->py,temp->pz);
            temp=temp->link;
           }
           printf("\n");
}



void eval( )
   {
int sum=0,x,y,z;
NODE head,temp;
head=(NODE)malloc(sizeof(struct poly));
head->link=head;
 printf("\nEnter polynomial to Evaluate\n");
 read(head);
 printf("Polynomial : ");
 display(head);
 printf("\n Enter the value of x,y and z \n");
 scanf("%d%d%d",&x,&y,&z);

 temp=head->link;
  while(temp!=head)
       {
        sum=sum+((temp->cf)*pow(x,temp->px)*
                        pow(y,temp->py)*pow(z,temp->pz));
        temp=temp->link;
       }
      printf("\n");
      printf("\n Sum=%d",sum);
      }

void polysum( )
  {
```

```
NODE temp,cur,prev,cur1,cur2,head1,head2;
int cf,px,py,pz,x,y,z;
head1=(NODE)malloc(sizeof(struct poly));
head2=(NODE)malloc(sizeof(struct poly));

head1->link=head1;
head2->link=head2;

    printf("\nEnter First polynomial");
    read(head1);

    printf("\nEnter Second polynomial");
    read(head2);

    printf("\nFirst polynomial : ");
    display(head1);

    printf("\nSecond polynomial : ");
    display(head2);



 cur1=head1->link;
  while(cur1!=head1)
      {
         prev=head2;
         cur2=head2->link;
         while(cur2!=head2)
             {
               if(cur1->px==cur2->px && cur1->py==cur2->py &&
                                                  cur1->pz==cur2->pz)
                  {
                   cur1->cf=cur1->cf+cur2->cf;
                   prev->link=cur2->link;
                  free(cur2);
                  break;
                 }
```

```
                    prev=cur2;
                    cur2=cur2->link;
                   }
                prev=cur1;
               cur1=cur1->link;
              }
        prev->link=head2->link;
        head2->link=head1->link;
       printf("\nResultant polynomial:");
      display(head2);
}
```

output:

```
1: Polynomial Evaluation
2: Sum of two polynomial
3: Exit

Enter choice
1

Enter polynomial to Evaluate
Enter no of terms
5
Enter cf,px,py and pz of 1 term
6 2 2 1
Enter cf,px,py and pz of 2 term
-4 0 1 5
Enter cf,px,py and pz of 3 term
3 3 1 1
Enter cf,px,py and pz of 4 term
2 1 5 1
Enter cf,px,py and pz of 5 term
-2 1 1 3

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

Enter the value of x,y and z
1 1 1

Sum = 5

Enter choice
2

Enter First polynomial
Enter no of terms
3
Enter cf,px,py and pz of 1 term
3 2 1 0
Enter cf,px,py and pz of 2 term
4 1 2 0
Enter cf,px,py and pz of 3 term
2 0 0 1

Enter Second polynomial
Enter no of terms
3
Enter cf,px,py and pz of 1 term
5 2 1 0
Enter cf,px,py and pz of 2 term
-4 1 2 0
Enter cf,px,py and pz of 3 term
3 0 0 2

First polynomial : + 3x^2y^1z^0 + 4x^1y^2z^0 + 2x^0y^0z^1
Second polynomial : + 5x^2y^1z^0 -4x^1y^2z^0 + 3x^0y^0z^2

Resultant polynomial:
+ 8x^2y^1z^0 + 2x^0y^0z^1 + 3x^0y^0z^2

1: Polynomial Evaluation
2: Sum of two polynomial

3: Exit

Enter choice
3

Process exited successfully.

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 in order, Preorder and Post Order
c. Search the BST for a given element (KEY) and report the appropriate message
d. Exit

Code:

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

struct BST
{
  int info;
  struct BST *llink,*rlink;
};
typedef struct BST *NODE;

void inorder(NODE  root)
{
 if(root==NULL)
 return;

 inorder(root->llink);
 printf("%d,",root->info);
 inorder(root->rlink);
}

void preorder(NODE  root)
{
 if(root==NULL)
 return;
```

```
 printf("%d,",root->info);
 preorder(root->llink);
 preorder(root->rlink);
}

void postorder(NODE  root)
{
 if(root==NULL)
 return;

 postorder(root->llink);
 postorder(root->rlink);
 printf("%d,",root->info);
}



void main( )
{
 NODE root=NULL,temp,cur,prev;
 int choice,item;
   while(1)
   {
     printf("\n\n Operations of Binary search tree \n");
     printf(" \n 1:create \n 2:traverse \n 3:Search key element\n 4:exit \n");
     printf("\n Enter your choice \n");
     scanf("%d",&choice);

     switch(choice)
           {
             case 1:temp=(NODE)malloc(sizeof(struct BST));
             printf("\n Enter an element \n");
             scanf("%d",&item);

             temp->info=item;
             temp->llink=temp->rlink=NULL;

             if(root==NULL)
             {
              root=temp;
              break;
```

```
}
prev=NULL;
cur=root;
if(item==cur->info)
  {
    printf("\n Dublicate items are not allowed \n");
    free(temp);
    break;
   }

while(cur!=NULL)
{
 prev=cur;
 if(item<cur->info)
   cur=cur->llink;
 else
   cur=cur->rlink;
 }
if(item<prev->info)
  prev->llink=temp;
else
  prev->rlink=temp;
  break;

  case 2:if(root==NULL)
           {
             printf("\n Tree is empty \n");
             break;
           }

printf("\n Contents of the tree are \n");
printf("\n Inorder :");
inorder(root);

printf("\n Preorder :");
preorder(root);

printf("\n Postorder :");
postorder(root);
break;
```

```
            case 3:if(root==NULL)
                    {
                      printf("\n Tree is empty \n");
                      break;
                    }
                   printf("\n Enter key element \n");
          scanf("%d",&item);

          cur=root;
          while(cur!=NULL)
            {
             if(item==cur->info)
                {
                   printf("\n Element is found \n");
                   break;
                 }
              if(item<cur->info)
                cur=cur->llink;
             else
               cur=cur->rlink;
            }
          if(cur==NULL)
          printf("\n Element is not found \n");
          break;

             case 4:exit(0);
           }
       }
}
```

Output:

Operations of Binary search tree

1: create
2: traverse
3: Search key element
4: exit

```
Enter your choice
1
Enter an element
50

Enter your choice
1
Enter an element
30

Enter your choice
1
Enter an element
70

Enter your choice
1
Enter an element
20

Enter your choice
1
Enter an element
40

Enter your choice
1
Enter an element
60

Enter your choice
1
Enter an element
80

Enter your choice
2
Contents of the tree are

Inorder :20,30,40,50,60,70,80,
Preorder :50,30,20,40,70,60,80,
```

Postorder :20,40,30,60,80,70,50,

Enter your choice
3
Enter key element
60
Element is found

Enter your choice
3
Enter key element
100
Element is not found

Enter your choice
4

11. Design, Develop and Implement a Program in C for the following operations on Graph(G) of Cities
a. Create a Graph of N cities using Adjacency Matrix.
b. Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method.

Code:

```
#include<stdio.h>

 int a[10][10],n,s[10];


void dfs(int  u)

{

  int v,i;

  s[u]=1;

  printf("%d",u);

  for(v=0;v<n;v++)

      {

        if(a[u][v]==1 && s[v]==0)

        dfs(v);

       }
```

```
}

 void main( )
  {
 int i,j,source;
 printf("Enter no of nodes in the graph \n");
 scanf("%d",&n);

 printf("Enter adjacency matrix \n");
 for(i=0;i<n;i++)
 for(j=0;j<n;j++)
 scanf("%d",&a[i][j]);

 printf("Enter source node in the graph \n");
 scanf("%d",&source);

 printf("\n The nodes reachable from %d:",source);
 dfs(source);
 printf("\n");
  }
```

output:

```
    Enter no of nodes in the graph
2
Enter adjacency matrix
```

```
1

2

3

4

Enter source node in the graph

2


 The nodes reachable from 2:2
```

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**.

Code:

```
#include<stdio.h>
#define  Hash_Size 10
int HT[Hash_Size];
main( )
{
   FILE *fp;
   int i,key,hv,index;
   char name[20];
   for(i=0;i< Hash_Size;i++)
   HT[i]=-1;
   fp=fopen("1.txt","r");
   while(!feof(fp))
     {
        fscanf(fp," %d %s ",&key,name);
        hv=key% Hash_Size;
         if(HT[hv]==-1)
          HT[hv]=key;
       else
           {
```

```
            printf("\ncollision for key %d:\n",key);
            for(i=0;i< Hash_Size;i++)
              {
                index=(hv+i)% Hash_Size;
                if(HT[index]==-1)
                  {
                    HT[index]=key;
                    printf("\ncollision solved using linear probing\n");
                    break;
                  }
              }
        }
    }
    if(i== Hash_Size)
    printf("hash table is full");
    printf("-------------\n");
    printf("hash table\n");
    printf("---------\n");
    printf("adress\tkey\n");
    for(i=0; i< Hash_Size; i++)
    printf("%d\t %d\n",i,HT[i]);
}
```

1.txt

1 2 3 4 5

3 4 2 1 5

4 3

3 4 3

3 3

2 43

4 42

23

Output

Collision for key 1 at address 1

Collision solved using linear probing at 2

Collision for key 4 at address 4

Collision solved using linear probing at 6

Collision for key 3 at address 3

Collision solved using linear probing at 7

Collision for key 3 at address 3

Collision solved using linear probing at 8

Collision for key 3 at address 3

Collision solved using linear probing at 9

Collision for key 43 at address 3

Collision solved using linear probing at 0

Collision for key 42 at address 2

Hash table is full cannot insert key 42

Hash Table

Address Key

0 43

1	1
2	1
3	3
4	4
5	5
6	4
7	3
8	3
9	3