

DATA STRUCTURES LABORATORY- BCSL305
SEM- III CSE

Program 1: Calendar Application.

Develop a Program in C for the following:

- 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>
#include <string.h>
```

```
struct Day {
    char *name;
    int date;
    char *activity;
};
```

```
void create(struct Day calendar[7])
{
    for (int i = 0; i < 7; i++)
    {
        calendar[i].name = (char *)malloc(20 * sizeof(char));
        printf("Enter day name: ");
        scanf("%s", calendar[i].name);
        printf("Enter date: ");
        scanf("%d", &calendar[i].date);
        calendar[i].activity = (char *)malloc(100 * sizeof(char));
    }
}
```

```
void read(struct Day calendar[7])
{
    printf("Enter the activity for each day of the week:\n");
    for (int i = 0; i < 7; i++)
    {
        printf("Day %s: %d - ", calendar[i].name, calendar[i].date);
        //gets(calendar[i].activity);
        scanf("%[^\n]s", calendar[i].activity);
    }
}
```

```
void display(struct Day calendar[7])
{

```

```

        printf("Week Activity Details:\n");
        for (int i = 0; i < 7; i++)
        {
            printf("Day: %s, Date: %d, Activity: %s\n", calendar[i].name, calendar[i].date,
calendar[i].activity);
        }
    }

int main()
{

    struct Day *calendar = (struct Day *) malloc(7 * sizeof(struct Day));
    if (calendar == NULL)
    {
        printf("Memory allocation failed.\n");
        return 1;
    }
    create(calendar);

    read(calendar);
    display(calendar);
    for (int i = 0; i < 7; i++)
    {
        free(calendar[i].name);
        free(calendar[i].activity);
    }
    free(calendar);
    return 0;
}

```

Output:

```
C:\Users\Arzoo>a
Enter day name: Sunday
Enter date: 11
Enter day name: Monday
Enter date: 12
Enter day name: Tuesday
Enter date: 13
Enter day name: Wednesday
Enter date: 14
Enter day name: Thursday
Enter date: 15
Enter day name: Friday
Enter date: 16
Enter day name: Saturday
Enter date: 17
Enter the activity for each day of the week:
Day Sunday: 11 - Outing with Family
Day Monday: 12 - Meeting with Guide
Day Tuesday: 13 - Entrance exam preparation
Day Wednesday: 14 - Coding Classes
Day Thursday: 15 - Entrance Preparation
Day Friday: 16 - Going to Gym
Day Saturday: 17 - Going to Gym
Week Activity Details:
Day: Sunday, Date: 11, Activity: Outing with Family
Day: Monday, Date: 12, Activity: Meeting with Guide
Day: Tuesday, Date: 13, Activity: Entrance exam preparation
Day: Wednesday, Date: 14, Activity: Coding Classes
Day: Thursday, Date: 15, Activity: Entrance Preparation
Day: Friday, Date: 16, Activity: Going to Gym
Day: Saturday, Date: 17, Activity: Going to Gym
```

Program 2. String Pattern Matching

Design, Develop and Implement 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>
```

```
char str[100], pat[20], rep[20], newstr[100];  
int i = 0, j = 0, k, n = 0, exist = 0, l=0;
```

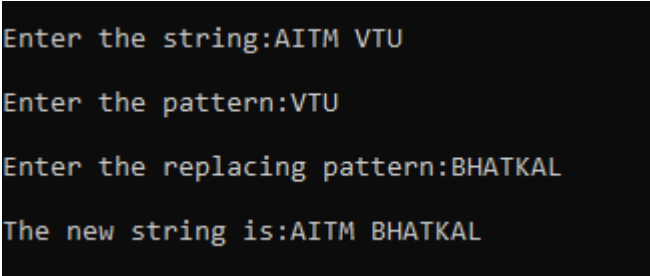
```
void matchAndReplace ();
```

```
int main( )  
{  
    printf("\nEnter the string:");  
    gets(str);  
    printf("\nEnter the pattern:");  
    gets(pat);  
    printf("\nEnter the replacing pattern:");  
    gets(rep);  
    matchAndReplace( );  
    return 0;  
}
```

```
void matchAndReplace( )  
{  
    while (str[i] != '\0')  
    {  
        j = 0, k = i;  
  
        while (str[k] == pat[j] && pat[j] != '\0')  
        {  
            k++;  
            j++;  
        }  
  
        if (pat[j] == '\0')  
        {  
            exist = 1;  
  
            for (int l = 0; rep[l] != '\0'; l++)  
                newstr[n++] = rep[l];  
            i=k;  
        }  
    }
```

```
        else
            newstr[n++] = str[i++];
    }
    if (!exist)
        printf("\nPattern does not exist");
    else
    {
        newstr[n] = '\0';
        printf("\nThe new string is:");
        puts(newstr);
    }
}
```

Output:

A screenshot of a terminal window with a black background and light blue text. It shows the output of a C program. The first line is 'Enter the string:AITM VTU'. The second line is 'Enter the pattern:VTU'. The third line is 'Enter the replacing pattern:BHATKAL'. The fourth line is 'The new string is:AITM BHATKAL'.

```
Enter the string:AITM VTU
Enter the pattern:VTU
Enter the replacing pattern:BHATKAL
The new string is:AITM BHATKAL
```

Program 3: Stack Operations

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

#define max 20
int s[max],top=-1,ch;
int i,temp,elem;
void push();
void pop();
void display();
void pali();

int main()
{
while(1)
{
printf("\n\n-----STACK OPERATIONS-----\n");
printf("1.Push\n2.Pop\n3.Display\n4.Palindrome\n5.Exit\n");
printf("\nEnter choice:");
scanf("%d",&ch);
switch(ch)
{
case 1: push();
break;
case 2: pop();
break;
case 3: display();
break;
case 4: pali();
break;
case 5:
exit(0);
}
}
}

void push()
{
printf("\nEnter the element to be inserted:");
```

```

        scanf("%d",&elem);
        if(top==max-1)
            printf("\nStack overflow");
        else
            s[++top]=elem;
        temp=top;
    }
void pop()
{
    if(top==-1)
        printf("\nStack underflow");
    else;
        top--;
}

void display()
{
    if(top==-1)
        printf("\nStack underflow");
    else
    {
        printf("\n Stack elements are:");
        for(i=0;i<=top;i++)
            printf("%d\t",s[i]);
    }
}

void pali()
{
    printf("\nEnter a number to check for palindrome:");
    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("\nNot a palindrome");
            return;
        }
        temp=temp/10;
    }
    printf("\nIt is a palindrome");
}

```

```

-----STACK OPERATIONS-----
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit

Enter choice:1

Enter the element to be inserted:10

-----STACK OPERATIONS-----
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit

Enter choice:1

Enter the element to be inserted:11

-----STACK OPERATIONS-----
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit

Enter choice:3

Stack elements are:10 11

```

```

-----STACK OPERATIONS-----
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit

Enter choice:4

Enter a number to check for palindrome:121

It is a palindrome

-----STACK OPERATIONS-----
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit

Enter choice:4

Enter a number to check for palindrome:12

Not a palindrome

-----STACK OPERATIONS-----
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit

Enter choice:5

```

```

-----STACK OPERATIONS-----
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit

Enter choice:2

-----STACK OPERATIONS-----
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit

Enter choice:3

Stack elements are:10

```


Program 4: Infix to Postfix using Stack.

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<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;
    }
}
```

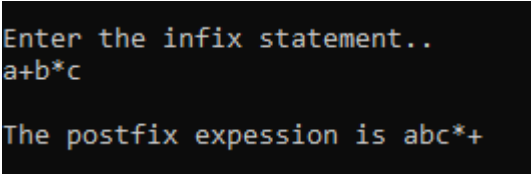
```
int main( )
{
    char s[20],infix[100],postfix[100];
```

```

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 expression is %s \n",postfix);
}

```

Output:



```

Enter the infix statement..
a+b*c

The postfix expression is abc*+

```

Program 5: Evaluation of Postfix Expression

Develop a Program in C for the following Stack Applications:

- a. Evaluate a Suffix-expression with single digit operands and operators: +, -, *, /, %, ^.

```
#include<stdio.h>
#include<string.h>
#include<math.h>
int main()
{
    char postfix[20];
    int result, i, s[20], op1,op2, top=-1;
    printf("\nEnter the postfix expression:");
    scanf("%s",postfix);
    for(i=0;i<strlen(postfix);i++)
    {
        if(isdigit(postfix[i])) // if operand push on stack

            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);
                    break;
            }
            s[++top]=result;
        }
    }
    printf("\nResult=%d",result);
}
```

Output:

```
Enter the postfix expression:235*+
Result=17
```

b. Solving Tower of Hanoi problem with n disks.

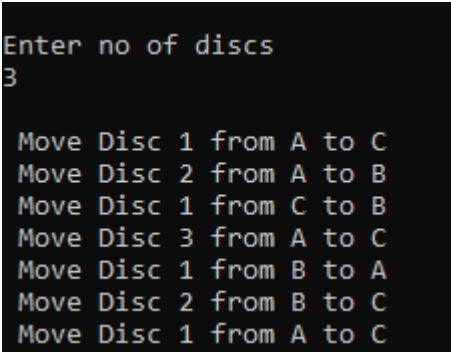
```
#include<stdio.h>

tower(int n, char s, char d, char t)
{
    if(n==0)
        return;

    tower(n-1,s,t,d);
    printf("\n Move Disc %d from %c to %c", n,s,d);
    tower(n-1,t,d,s);
}

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

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
```

Program 6: Circular Queue

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 5

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

char q[MAX], elem;
int f=0, r=0, i, c=0, front, ch;
void insert()
{
    if(c==MAX)
    {
        printf("\nQueue is full");
        return;
    }
    printf("Enter the character to insert:");
    scanf("%s",&elem);
    q[r]=elem;
    r=(r+1)%MAX;
    c++;
}

void delete()
{
    if (c == 0)
        printf("Queue is empty");
    else
    {
        printf("\nDeleted element is:%c",q[f]);
        f = (f+1)% MAX;
        c--;
    }
}

void display()
{
    if(c==0)
```

```

{
    printf("Queue is empty\n");
    return;
}

printf("\nQueue elements are:");
front=f;
for(i=0; i<c;i++)
{
    printf("%c\t",q[front]);
    front=(front+1)%MAX;
}
printf("\n");
}

int main()
{
while(1)
{
    printf("\n*Circular Queue Operations*\n1.Insert\n2.Delete\n3.Display\n4.Exit");
    printf("\nEnter your choice:");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1: insert();
                break;
        case 2: delete();
                break;
        case 3: display();
                break;
        case 4: exit(0);
                }
    }
}
}

```

Output:

Circular Queue Operations

- 1.Insert
- 2.Delete
- 3.Display
- 4.Exit

Enter your choice:1

Enter the character to insert:A

Circular Queue Operations

- 1.Insert
- 2.Delete
- 3.Display
- 4.Exit

Enter your choice:1

Enter the character to insert:B

Circular Queue Operations

- 1.Insert
- 2.Delete
- 3.Display
- 4.Exit

Enter your choice:2

Deleted element is:A

Circular Queue Operations

- 1.Insert
- 2.Delete
- 3.Display
- 4.Exit

Enter your choice:3

Queue elements are:B

Circular Queue Operations

- 1.Insert
- 2.Delete
- 3.Display
- 4.Exit

Enter your choice:4

Program 7. Singly Linked List Operations.

Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Programme, Sem, PhNo

- Create a SLL of N Students Data by using front insertion.
- Display the status of SLL and count the number of nodes in it
- Perform Insertion / Deletion at End of SLL
- Perform Insertion / Deletion at Front of SLL(Demonstration of stack)
- Exit

```
#include<stdio.h>
#include<stdlib.h>
struct student
{
    char name[15],sem[15],usn[15],branch[15],phone[15];
    struct student *link;
};
struct student *first=NULL,*temp,*cur,*prev;

void insertf();
void insertr();
void deletef();
void deleter();
void display();

int main( )
{
    int choice;
    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:insertf();
            break;
            case 2:insertr();
            break;
            case 3:deletef();
            break;
            case 4:deleter();
            break;
            case 5:display();
            break;
            case 6:exit(0);
        }
    }
}
```



```

void insertf()
{
    temp=(struct student*) malloc(sizeof(struct student));
    printf("\nEnter the name,sem,usn,branch and phone.no:");
    scanf("%s%s%s%s%s",temp->name,temp->sem,temp->usn,temp->branch,temp->phone);
    temp->link=first;
    first=temp;
}

void insertr()
{
    temp=(struct student*) malloc(sizeof(struct student));
    printf("\nEnter the name,sem,usn,branch and phone.no:");
    scanf("%s%s%s%s%s",temp->name,temp->sem,temp->usn,temp->branch,temp->phone);
    temp->link=NULL;
    if(first==NULL)
    {
        first=temp;
        return;
    }
    cur=first;
    while(cur->link!=NULL)
        cur=cur->link;
    cur->link=temp;
}

void display( )
{
    int c=0;
    temp=(struct student*) malloc(sizeof(struct student));
    if(first==NULL)
    {
        printf("\n list is empty \n");
        return;
    }
    printf("\n The contents of linked list are \n");
    printf("name\ttsem\ttusn\ttBranch\ttPhone no ");
    printf("\n.....\n");
    temp=first;
    while(temp!=NULL)
    {
        printf("%s\t\t%s\t\t%s\t\t%s\t\t%s",temp->name,temp->sem,temp->usn,temp->branch,temp->phone);
        printf("\n");
        temp=temp->link;
        c++;
    }
    printf("\nTotal no.of Students:%d",c);
}

```

```

void deletetf()
{
    if(first==NULL)
    {
        printf("\n List is empty \n");
        return;
    }
    temp=first;
    printf("\n First student details in the list is deleted\n");
    first=first->link;
    free(temp);
}

```

```

void deleter()
{
    if(first==NULL)
    {
        printf("\n List is empty \n");
        return;
    }
    if(first->link==NULL)
    {
        printf(" List contains one student details,that is deleted");
        first=NULL;
        return;
    }
    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);
}

```

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 the name,sem,usn,branch and phone.no:Amit 3 001 cse 6789012345

***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 the name,sem,usn,branch and phone.no:Arun 3 002 cse 7878923471

***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
5

The contents of linked list are
name          sem          usn          Branch          Phone no
.....
Arun          3          002          cse          7878923471

Total no.of Students:1
***Operations of Singly linked list***

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:display
6:exit
```

Program.8. DLL

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.

- Create a DLL of N Employees Data by using end insertion.
- Display the status of DLL and count the number of nodes in it
- Perform Insertion and Deletion at End of DLL
- Perform Insertion and Deletion at Front of DLL
- Demonstrate how this DLL can be used as Double Ended Queue.
- Exit

```
#include<stdio.h>
#include<stdlib.h>
struct employee
{
char name[15],ssn[15],dept[15],des[15],sal[15],phno[15];
struct employee *rlink,*llink;
};
struct employee *first=NULL,*temp,*cur,*prev;

void insertf();
void insertr();
void deletef();
void deleter();
void display();

int main( )
{
int choice;
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:insertf();
break;
case 2:insertr();
break;
case 3:deletef();
break;
case 4:deleter();
break;
case 5:display();
break;
case 6:exit(0);
}
}
}
```

```

}

void insertf()
{
temp=(struct employee*) malloc(sizeof(struct employee));
printf("\nEnter the name,ssn,dept,designation,salary and phone.no:");
scanf("%s%s%s%s%s%s",temp->name,temp->ssn,temp->dept,temp->des,temp->sal,temp->phno);
temp->llink=temp->rlink=NULL;
if(first==NULL)
{
    first=temp;
    return;
}

temp->rlink=first;
first->llink=temp;
first=temp;
}

void insertr()
{
temp=(struct employee*) malloc(sizeof(struct employee));
printf("\nEnter the name,ssn,dept,designation,salary and phone.no:");
scanf("%s%s%s%s%s%s",temp->name,temp->ssn,temp->dept,temp->des,temp->sal,temp->phno);
temp->llink=temp->rlink=NULL;
if(first==NULL)
{
    first=temp;
    return;
}
cur=first;
while(cur->rlink!=NULL)
    cur=cur->rlink;
cur->rlink=temp;
temp->llink=cur;
}

void display( )
{
int c=0;
temp=(struct employee*) malloc(sizeof(struct employee));
if(first==NULL)
{
    printf("\n list is empty \n");
    return;
}
printf("\n The contents of linked list are \n");
printf("Name\t\tSSN\t\tDept\t\tDesignation\t\tSalary\t\tPhoneNo ");
printf("\n.....\n");

```

```

temp=first;
while(temp!=NULL)
{
    printf("%s\t%s\t%s\t%s\t%s\t%s\t%s",temp->name,temp->ssn,temp->dept,temp->des,temp->
sal,temp->phno);
    printf("\n");
    temp=temp->rlink;
    c++;
}
printf("\nTotal no.of Students:%d",c);
}

```

```

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

```

```

void deleter()
{
if(first==NULL)
{
    printf("\n List is empty \n");
    return;
}
if(first->rlink==NULL)
{
    printf(" List contains only one node,that is deleted");
    first=NULL;
    return;
}
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);

```

}

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 the name,ssn,dept,designation,salary and phone.no:John 334 Finance Manager 75000 7685432198

***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
```

Program 9: Evaluation of Polynomial:

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.

```
#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
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 2 1

Sum=82
1:Polynomial Evaluation
2:Sum of two polynomial
3:exit

Enter choice
2

Enter First polynomial
Enter no of terms
2

Enter cf,px,py and pz of 1 term
2
1
1
1

Enter cf,px,py and pz of 2 term
5
2
2
2

Enter Second polynomial
Enter no of terms
2

Enter cf,px,py and pz of 1 term
1
1
1
1

Enter cf,px,py and pz of 2 term
3
2
2
2
```

```
First polynomial : + 2x^1y^1z^1 + 5x^2y^2z^2
Second polynomial : + 1x^1y^1z^1 + 3x^2y^2z^2
Resultant polynomial: + 3x^1y^1z^1 + 8x^2y^2z^2

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

Program 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 BST
{
int info;
struct BST *llink,*rlink;
};

struct BST *root=NULL,*temp,*cur,*prev;
int item;
void create();
void inorder();
void preorder();
void postorder();
void search();

void main( )
{
int choice;
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:create();
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:search();
        break;
        case 4:exit(0);
    }
}
}

void create()
{
temp=(struct BST*)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;
return;
}

prev=NULL;
cur=root;
while(cur!=NULL)
{
    if(item==cur->info)
    {
        printf("\n Duplicate items are not allowed \n");
        free(temp);
        return;
    }
    prev=cur;
    if(item<cur->info)
        cur=cur->llink;
    else
        cur=cur->rlink;
}

    if(item<prev->info)
        prev->llink=temp;
    else
        prev->rlink=temp;
}

void inorder(struct BST *root)
{
if(root==NULL)
    return;
inorder(root->llink);
printf("%d,",root->info);

```

```
inorder(root->rlink);
}
```

```
void preorder(struct BST *root)
{
if(root==NULL)
    return;
printf("%d,",root->info);
preorder(root->llink);
preorder(root->rlink);
}
```

```
void postorder(struct BST *root)
{
if(root==NULL)
    return;
postorder(root->llink);
postorder(root->rlink);
printf("%d,",root->info);
}
```

```
void search()
{
if(root==NULL)
{
    printf("\n Tree is empty \n");
    return;
}
printf("\n Enter key element \n");
scanf("%d",&item);
cur=root;
while(cur!=NULL)
{
    if(item==cur->info)
    {
        printf("\n Element is found \n");
        return;
    }
    if(item<cur->info)
        cur=cur->llink;
    else
        cur=cur->rlink;
}
if(cur==NULL)
    printf("\n Element is not found \n");
}
```

Output:

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

Enter your choice
1

Enter an element
9

Operations of Binary search tree

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

Enter your choice
1

Enter an element
5

Operations of Binary search tree

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

Enter your choice
1

Enter an element
2
```

```
Enter your choice
2

Contents of the tree are

Inorder :2,5,6,7,8,9,14,15,24,
Preorder :6,5,2,9,8,7,15,14,24,
Postorder :2,5,7,8,14,24,15,9,6,

Operations of Binary search tree

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

Enter your choice
3

Enter key element
9

Element is found

Operations of Binary search tree

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

Enter your choice
3

Enter key element
89

Element is not found
```

11. DFS Method

Design, Develop and Implement a Program in C for the following operations on Graph(G) of Cities.

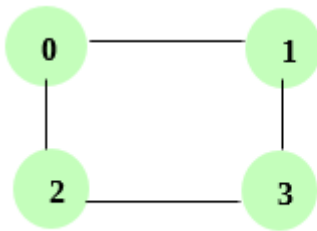
- Create a Graph of N cities using Adjacency Matrix.
- Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method.

```
#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);
    }
}

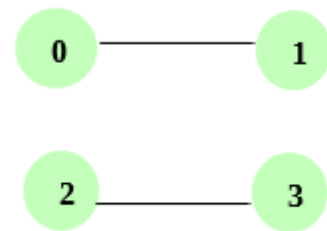
void main( )
{
    int i,j,source;
    printf("Enter no of nodes in the graph \n");
    scanf("%d",&n);
    printf("Enter adjacency matrix \n");
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            scanf("%d",&a[i][j]);
    printf("Enter source node in the graph \n");
    scanf("%d",&source);
    printf("\n The nodes reachable from %d:",source);
    dfs(source);
    printf("\n");
}
```


Output:



```
Enter no of nodes in the graph
4
Enter adjacency matrix
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
Enter source node in the graph
1
```

The nodes reachable from 1:1023



```
Enter no of nodes in the graph
4
Enter adjacency matrix
0 1 0 0
1 0 0 0
0 0 0 1
0 0 1 0
Enter source node in the graph
2
```

The nodes reachable from 2:23

Program 12. Hashing and Linear Probing.

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: $K \rightarrow L$ as $H(K)=K \bmod m$ (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>
#define hash_size 10
int ht[hash_size];

int main()
{
    FILE *fp;
    int i,key,hv,index;
    char name[20];
    for(i=0;i<hash_size;i++) // Initialize the hash table
        ht[i]=-1;
    fp=fopen("emp.txt","r"); // open file for reading
    while(!feof(fp)) // until eof
    {
        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("Collision solved used linear probing\n");
                    break;
                }
            }
        }
    }
    if(i==hash_size)
        printf("\nHash table is full\n");
    printf("-----\n");
    printf("\nHash Table\n");
    printf("-----\n");
    printf("Address\tKey\n");
```

```
for(i=0;i<hash_size;i++)
printf("%d\t%d\n",i,ht[i]);
fclose(fp);
return 0;
}
```

Output:

```
Collision for key1254:
Collision solved used linear probing
```

```
Collision for key1255:
Collision solved used linear probing
```

```
-----
```

```
Hash Table
```

```
-----
```

```
Address Key
```

```
0      1230
```

```
1      1231
```

```
2      -1
```

```
3      -1
```

```
4      1234
```


```
5      1235
```

```
6      1236
```

```
7      1254
```

```
8      1255
```

```
9      -1
```

 emp.txt - Notepad

File Edit Format View Help

1230 Riya

1231 Akash|

1234 Amit

1235 Mukesh

1236 Sushant

1254 Preethi

1255 Joseph