VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI - 590018, KARNATAKA

Lab Manual

On

"DATA STRUCTURES AND APPLICATION"

BCS305

III Semester of

Bachelor of Engineering in Computer Science and Engineering of

Visvesvaraya Technological University, Belagavi

Mrs .Pallavi K V

Assistant Professor

Dept. of Computer Science and Engineering

AMC Engineering College

Department of Computer Science and Engineering

AMC ENGINEERING COLLEGE

18 Km, Bannerghatta Road, Bangalore - 560083

2023-24

1. **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;
};
struct Day*  calendar[7];
void create( )
{
   int i;
   for(i=0;i<7;i++)
   {
      calendar[i]=(struct Day*)malloc(sizeof(struct Day));
      calendar[i]->name=(char*)malloc(20*sizeof(char));
      calendar[i]->activity=(char*)malloc(100*sizeof(char));
   }
```

```
}
void read()
{
   int i;
   for(i=0;i<7;i++)
   {
      printf("Enter day name:");
      scanf("%s",calendar[i]->name);
      printf("Enter Date:");
      scanf("%d",&calendar[i]->date);
      printf("Enter activity:");
      scanf("%s",calendar[i]->activity);
      printf("\n");
   }
}
void display()
{
   int i;
   printf("Weekly activity details:\n");
   for(i=0;i<7;i++)
   {
      printf("Day%d:%s--%d,\tActivity:%s\n",i+1,calendar[i]-
>name,calendar[i]->date,calendar[i]->activity);
   }
}
int main()
{
```

```
    int i;
    printf("Creating calendar:\n");
    create();
    printf("Reading calendar:\n");
    read();
    printf("Displaying calendar:\n");
    display();
    for(i=0;i<7;i++)
    {
       free(calendar[i]->name);
       free(calendar[i]->activity);
       free(calendar[i]);
    }
    return 0;
}
```

OUTPUT:

Creating calendar:

Reading calendar:

Enter day name: MONDAY

Enter Date:1

Enter activity: RUN

Enter day name: TUESDAY

Enter Date:2

Enter activity: JOG

Enter day name: WEDNESDAY

Enter Date:3

Enter activity: SWIM

Enter day name: THURSDAY

Enter Date:4

Enter activity: SLEEP

Enter day name: FRIDAY

Enter Date:5

Enter activity: COOK

Enter day name: SATURDAY

Enter Date:6

Enter activity: DANCE

Enter day name: SUNDAY

Enter Date:7

Enter activity: SING

Displaying calendar:

Weekly activity details:

Day1:MONDAY--1, Activity: RUN

Day2:TUESDAY--2, Activity: JOG

Day3:WEDNESDAY--3, Activity: SWIM

Day4:THURSDAY--4, Activity: SLEEP

Day5:FRIDAY--5, Activity: COOK

Day6:SATURDAY--6, Activity: DANCE

Day7:SUNDAY--7, Activity: SING

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 message in case PAT does not exist in STR.**

Support the program with functions for each of the above operations. Don't use built in functions.

```
#include<stdio.h>
void read_data();
void search_replace();
char str[100], pat[100], rep[100], update[100];
void main()
{
   read_data();
   search_replace();
}
void read_data()
{
   printf("\nEnter a string:\n");
   gets(str);
   printf("\nEnter a search string:\n");
   gets(pat);
   printf("\nEnter a replace string:\n");
   gets(rep);
}
void search_replace()
{
   int i=0, j=0, c=0, m=0, k, flag=0;
```

```
while(str[c]!='\0')
{
   if(str[m]==pat[i])
   {
      i++;
      m++;
      if(pat[i]=='\0')
      {
         flag=1;
         for(k=0;rep[k]!='\0';k++,j++)
            update[j]=rep[k];
         i=0;
         c=m;
      }
   }
   else
   {
      update[j]=str[c];
      j++;
      c++;
      m=c;
      i=0;
   }
}
if(flag==1)
{
```

```
        update[j]='\0';
        printf("The resultant string is \n%s", update);
    }
    else
        printf("string not found:");
}
```

OUTPUT :

Enter a string:

GOOD MORNING

Enter a search string:

MORN

Enter a replace string:

EVEN

The resultant string is

GOOD EVENING

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 situation 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 10
int s[MAX], item, top = -1;
void push()
{
   if (top == MAX - 1)
   {
      printf("Stack overflow\n");
      return;
   }
   top++;
   s[top] = item;
}
int pop()
{
   if (top == -1)
      return -1;
   return s[top--];
```

```
}
void display()
{
   if (top == -1)
   {
      printf("Stack is empty\n");
      return;
   }
   printf("Contents of the stack:\n");
   for (int i = 0; i <= top; i++)
   {
      printf("%d\n", s[i]);
   }
}
void palindrome()
{
   int i, j;
   for (i = 0, j = top; i <= j; i++, j--) {
      if (s[i] != s[j])
      {
         printf("Not palindrome\n");
         return;
      }
   }
   printf("Palindrome\n");
}
```

```
int main()
{
   int ch;
   while (1)
   {
      printf("MENU\n");
      printf("1.Push\t2.Pop\t3.Check palindrome\t4.Display\t5.Exit\n");
      printf("--------------------\n");
      printf("Enter your choice:\n");
      scanf("%d", &ch);
      switch (ch) {
      case 1:
         printf("Enter the item:");
         scanf("%d", &item);
         push();
         break;
      case 2:
         printf("Popped value=%d\n", pop());
         break;
      case 3:
         palindrome();
         break;
      case 4:
         display();
         break;
      case 5:
```

```
            exit(0);
        default:
            printf("Invalid choice\n");
            break;
        }
    }
    return 0;
}
```

OUTPUT :

MENU

1.Push 2.Pop 3.Check palindrome 4.Display 5.Exit

Enter your choice:

4

Stack is empty

MENU

1.Push 2.Pop 3.Check palindrome 4.Display 5.Exit

Enter your choice:

1

Enter the item:1

MENU

1.Push 2.Pop 3.Check palindrome 4.Display 5.Exit

Enter your choice:

1

Enter the item:2

MENU

1.Push 2.Pop 3.Check palindrome 4.Display 5.Exit

Enter your choice:

1

Enter the item:1

MENU

1.Push 2.Pop 3.Check palindrome 4.Display 5.Exit

Enter your choice:

3

Palindrome

MENU

1.Push 2.Pop 3.Check palindrome 4.Display 5.Exit

Enter your choice:

4

Contents of the stack:

1

2

1

MENU

1.Push 2.Pop 3.Check palindrome 4.Display 5.Exit

Enter your choice: 5

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>
int F(char symbol)
{
   switch(symbol)
   {
      case '+':
      case '-':
            return 2;
      case '*':
      case '%':
      case '/':
            return 4;
      case '^':
      case '$':
            return 5;
      case '(':
            return 0;
      case '#':
            return -1;
      default : return 8;
   }
```

```
}
int G(char symbol)
{
   switch(symbol)
   {
      case '+':
      case '-':
            return 1;
      case '*':
      case '%':
      case '/':
            return 3;
      case '^':
      case '$':
            return 6;
      case '(':
            return 9;
      case '#':
            return 0;
      default : return 7;
   }
}
void infix_postfix(char infix[],char postfix[])
{
   int top,i,j;
   char s[30];
```

```
    char symbol;
    top=-1;
    s[++top]='#';
    j=0;
    for(i=0;i<strlen(infix);i++)
    {
        symbol=infix[i];
        while(F(s[top])>G(symbol))
        postfix[j++]=s[top--];
        if(F(s[top])!=G(symbol))
            s[++top]=symbol;
        else top--;
    }
    while(s[top]!='#')
    postfix[j++]=s[top--];
    postfix[j]='\0';
}
void main()
{
    char infix[20];
    char postfix[20];
    printf("Enter the valid infix expression:\n");
    scanf("%s",infix);
    infix_postfix(infix,postfix);
    printf("The postfix expression is:\n");
    printf("%s\n",postfix);
```

```
}
```

OUTPUT :

Enter the valid infix expression:

A-B-D*E/F+B*C

The postfix expression is:

AB-DE*F/-BC*+

5. Develop a program in C for the following stack application

a) Evaluation of suffix expression with single digit operands and operators + , - , * , / , % , ^ .

b) Solving Tower of Hanoi with n disks.

a) Evaluation of suffix expression with single digit operands and operators + , - , * , / , % , ^ .

```
#include<stdio.h>
#include<string.h>
#include<math.h>
double compute(char symbol, double op1, double op2)
{
   switch(symbol)
   {
      case '+':
            return op1+op2;
            break;
      case '-':
            return op1-op2;
            break;
      case '*':
            return op1*op2;
            break;
      case '/':
            return op1/op2;
            break;
      case '$':
      case '^':
```

```
            return pow(op1,op2);
            break;
      default:return 0;
   }
}
void main()
{
   double s[20],res,op1,op2;
   int top,i;
   char postfix[20],symbol;
   printf("\nEnter the postfix expression:\n");
   gets(postfix);
   top=-1;
   for(i=0;i<strlen(postfix);i++)
   {
      symbol=postfix[i];
      if(isdigit(symbol))
         s[++top]=symbol-'0';
      else
      {
         op2=s[top--];
         op1=s[top--];
         res=compute(symbol,op1,op2);
         s[++top]=res;
      }
   }
```

```
   res=s[top--];
   printf("\nThe result is :%f\n",res);
}
```

OUTPUT :

Enter the postfix expression:

65+3-

The result is :8.000000

b) Solving Tower of Hanoi with n disks.

```
#include<stdio.h>
void transfer(int n,char s,char t,char d)
{
   if(n==0)
      return;
   transfer(n-1,s,d,t);
   printf("Move disc %d from %c to %c\n",n,s,d);
   transfer(n-1,t,s,d);
}
void main()
{
   int n;
   printf("Enter the number of discs:\n");
   scanf("%d",&n);
   transfer(n,'A','B','C');
}
```

OUTPUT :

Enter the number 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. Develop a menu driven Program in C for the following operations on Circular QUEUE of

a) **Characters (Array Implementation of Queue with maximum size MAX)**
b) **Insert an Element on to Circular QUEUE**
c) **Delete an Element from Circular QUEUE**
d) **Demonstrate Overflow and Underflow situations on Circular QUEUE**
e) **Display the status of Circular QUEUE**
f) **Exit**

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

```
#include<stdio.h>
#include<stdlib.h>
#define QUE_SIZE 5
char item;
int count, q[QUE_SIZE], front, rear;
void insertQ()
{
   if (count == QUE_SIZE)
   {
      printf("Queue Overflow\n");
      return;
   }
   rear = (rear + 1) % QUE_SIZE;
   q[rear] = item;
   count++;
}
char deleteQ()
{
   if (count == 0)
```

```
        return -1;
    int item = q[front];
    front = (front + 1) % QUE_SIZE;
    count--;
    return item;
}
void display()
{
    if (count == 0)
    {
        printf("Queue is empty\n");
        return;
    }
    printf("Contents of the queue are:\n");
    int i, f;
    for (i = 0, f = front; i < count; i++)
    {
        printf("%c\n", q[f]);
        f = (f + 1) % QUE_SIZE;
    }
}
int main()
{
    int choice;
    front = 0;
    rear = -1;
```

```
for (;;)
{
   printf("1.Insert\t 2.Delete\n");
   printf("3.Display\t 4.Exit\n");
   printf("Enter your choice: ");
   scanf("%d", &choice);
   switch (choice)
   {
      case 1:
         printf("Enter the item to be inserted: ");
         scanf(" %c", &item);
         insertQ();
         break;
      case 2:
         item = deleteQ();
         if (item == -1)
         {
            printf("Queue is empty\n");
         }
         else
         {
            printf("Item deleted from queue is %c\n", item);
         }
         break;
      case 3:
         display();
```

```
            break;
        case 4:
            exit(0);
        default:
            printf("Invalid choice\n");
      }
   }
   return 0;
}
```

OUTPUT:

1.Insert 2.Delete

3.Display 4.Exit

Enter your choice: 1

Enter the item to be inserted: A

1.Insert 2.Delete

3.Display 4.Exit

Enter your choice: 1

Enter the item to be inserted: B

1.Insert 2.Delete

3.Display 4.Exit

Enter your choice: 1

Enter the item to be inserted: C

1.Insert 2.Delete

3.Display 4.Exit

Enter your choice: 1

Enter the item to be inserted: D

1.Insert 2.Delete

3.Display 4.Exit

Enter your choice: 1

Enter the item to be inserted: E

1.Insert 2.Delete

3.Display 4.Exit

Enter your choice: 3

Contents of the queue are:

A

B

C

D

E

1.Insert 2.Delete

3.Display 4.Exit

Enter your choice: 1

Enter the item to be inserted: F

Queue Overflow

1.Insert 2.Delete

3.Display 4.Exit

Enter your choice: 2

Item deleted from queue is A

1.Insert 2.Delete

3.Display 4.Exit

Enter your choice: 3

Contents of the queue are:

B

C

D

E

1.Insert 2.Delete

3.Display 4.Exit

Enter your choice: 4

7. Develop a menu driven program in C for the following operations on singly linked lists(SLL) of student data with the fields : USN, Name , Program , Sem, Ph no.

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 the end of SLL.
d) Perform insertion / deletion at the front of SLL.
e) Exit.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node
{
   int USN, SEM, PHNO;
   char NAME[20], BRANCH[10];
   struct node *link;
};

typedef struct node *NODE;
NODE first = NULL;
int count = 0;

int USN, SEM, phno;
char name[20], branch[10];

NODE insert_front()
{
   NODE temp;
   temp = (NODE)malloc(sizeof(struct node));
   printf("Enter student details:\n");
   printf("USN\t Name\tBranch\tSem\tPhno.\n");
   scanf("%d\t%s\t%s\t%d\t%d", &USN, name, branch, &SEM, &phno);
   temp->USN = USN;
   strcpy(temp->NAME, name);
   strcpy(temp->BRANCH, branch);
   temp->SEM = SEM;
```

```
    temp->PHNO = phno;
    temp->link = first;
    count++;
    return temp;
}

void create()
{
    int i, n;
    printf("Enter number of students:\n");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        first = insert_front();
    }
}

NODE delete_front()
{
    NODE temp;
    if (first == NULL)
    {
        printf("List is empty cannot delete\n");
        return first;
    }
    temp = first;
    first = first->link;
    printf("Student usn=%d,name=%s\n", temp->USN, temp->NAME);
    free(temp);
    count--;
    return first;
}

NODE insert_rear()
{
    NODE temp;
    NODE cur;
    temp = (NODE)malloc(sizeof(struct node));
```

```
    printf("Enter student details:\n");
    printf("USN\tName\tBranch\tSem\tPhno\n");
    scanf("%d %s %s %d %d", &USN, name, branch, &SEM, &phno);
    temp->USN = USN;
    strcpy(temp->NAME, name);
    strcpy(temp->BRANCH, branch);
    temp->SEM = SEM;
    temp->PHNO = phno;
    temp->link = NULL;

    if (first == NULL)
    {
        first = temp;
        count++;
        return first;
    }

    cur = first;
    while (cur->link != NULL)
    {
        cur = cur->link;
    }
    cur->link = temp;
    count++;
    return first;
}

NODE delete_rear()
{
    NODE temp, cur, prev;
    if (first == NULL)
    {
        printf("List is empty cannot delete\n");
        return first;
    }
    if (first->link == NULL)
    {
        printf("Student USN=%d;name=%s\n", first->USN, first->NAME);
```

```
        count--;
        free(first);
        return NULL;
    }
    prev = NULL;
    cur = first;
    while (cur->link != NULL)
    {
        prev = cur;
        cur = cur->link;
    }
    printf("Student USN=%d;name=%s\n", cur->USN, cur->NAME);
    free(cur);
    prev->link = NULL;
    count--;
    return first;
}

void display()
{
    NODE temp;
    if (first == NULL)
    {
        printf("List is empty\n");
        return;
    }
    printf("The contents of singly linked list:\n");
    temp = first;
    while (temp != NULL)
    {
        printf("%d\t%s\t%s\t%d\t%d\n", temp->USN, temp->NAME, temp-
>BRANCH, temp->SEM, temp->PHNO);
        temp = temp->link;
    }
    printf("The number of nodes in singly linked list=%d\n", count);
}

int main()
```

```
{
   int choice;
   for (;;)
   {

printf("\n1.Create\t2.Display\n3.Insert_rear\t4.Delete_rear\n5.Demonstration
of stack Insert_front,Delete_front\nEnter your choice:\n");
      scanf("%d", &choice);
      switch (choice)
      {
      case 1:
         create();
         break;
      case 2:
         display();
         break;
      case 3:
         first = insert_rear();
         break;
      case 4:
         first = delete_rear();
         break;
      case 5:
         printf("Push\n");
         first = insert_front();
         printf("Pop\n");
         first = delete_front();
         break;
      default:
         printf("INVALID CHOICE\n");
         exit(0);
      }
   }
   return 0;
}
```

OUTPUT :

```
1.Create      2.Display
3.Insert_rear      4.Delete_rear
5.Demonstration of stack Insert_front,Delete_front
Enter your choice:
1
Enter number of students:
3
Enter student details:
USN   Name      Branch      Sem   Phno.
1 XXX CSE 3 100
Enter student details:
USN   Name      Branch      Sem   Phno.
2 YYY CSE 3 101
Enter student details:
USN   Name      Branch      Sem   Phno.
3 ZZZ CSE 3 102
1.Create      2.Display
3.Insert_rear      4.Delete_rear
5.Demonstration of stack Insert_front,Delete_front
Enter your choice:
2
The contents of singly linked list:
3       ZZZ   CSE   3       102
2       YYY   CSE   3       101
1       XXX   CSE   3       100
The number of nodes in singly linked list=3

1.Create      2.Display
3.Insert_rear      4.Delete_rear
5.Demonstration of stack Insert_front,Delete_front
Enter your choice:
3
Enter student details:
USN   Name Branch      Sem   Phno
0 XYZ CSE 3 099
1.Create      2.Display
3.Insert_rear      4.Delete_rear
```

5.Demonstration of stack Insert_front,Delete_front
Enter your choice:
2
The contents of singly linked list:
3 ZZZ CSE 3 102
2 YYY CSE 3 101
1 XXX CSE 3 100
0 XYZ CSE 3 99
The number of nodes in singly linked list=4

1.Create 2.Display
3.Insert_rear 4.Delete_rear
5.Demonstration of stack Insert_front,Delete_front
Enter your choice:
4
Student USN=0;name=XYZ

1.Create 2.Display
3.Insert_rear 4.Delete_rear
5.Demonstration of stack Insert_front,Delete_front
Enter your choice:
2
The contents of singly linked list:
3 ZZZ CSE 3 102
2 YYY CSE 3 101
1 XXX CSE 3 100
The number of nodes in singly linked list=3

1.Create 2.Display
3.Insert_rear 4.Delete_rear
5.Demonstration of stack Insert_front,Delete_front
Enter your choice:
5
Push
Enter student details:
USN Name Branch Sem Phno.
4 SGK CSE 3 988
Pop

Student usn=4,name=SGK

1.Create 2.Display
3.Insert_rear 4.Delete_rear
5.Demonstration of stack Insert_front,Delete_front
Enter your choice:
2
The contents of singly linked list:
3 ZZZ CSE 3 102
2 YYY CSE 3 101
1 XXX CSE 3 100
The number of nodes in singly linked list=3

1.Create 2.Display
3.Insert_rear 4.Delete_rear
5.Demonstration of stack Insert_front,Delete_front
Enter your choice: 0

8. Develop a menu driven program in C for the following operations on doubly linked lists(DLL) of employee data with the fields : SSN, Name , Dept , Designation, Sal, Ph no.

a) Create a DLL of n Employee data by using end insertion.
b) Display the status of DLL and count the number of nodes in it.
c) Perform insertion / deletion at the end of DLL.
d) Perform insertion / deletion at the front of DLL.
e) Demonstration how this DLL can be used as Double Ended Queue.
f) Exit.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct
{
   int ssn;
   char name[20], dept[20], desg[20], ph[20];
   float sal;
} EMPL;

struct node
{
   int ssn;
   char name[20], dept[20], desg[20], ph[20];
   float sal;
   struct node *llink;
   struct node *rlink;
};
typedef struct node *NODE;
NODE getnode()
{
   NODE x;
```

```
    x = (NODE)malloc(sizeof(struct node));
    if (x == NULL)
    {
        printf("Out of memory\n");
        exit(0);
    }
    return x;
}
NODE insert_front(EMPL item, NODE first)
{
    NODE temp;
    temp = getnode();
    temp->ssn = item.ssn;
    strcpy(temp->name, item.name);
    strcpy(temp->dept, item.dept);
    strcpy(temp->desg, item.desg);
    temp->sal = item.sal;
    strcpy(temp->ph, item.ph);
    temp->llink = temp->rlink = NULL;
    if (first == NULL)
        return temp;
    temp->rlink = first;
    first->llink = temp;
    return temp;
}
NODE insert_rear(EMPL item, NODE first)
{
    NODE temp, cur;
    temp = getnode();
```

```
    temp->ssn = item.ssn;
    strcpy(temp->name, item.name);
    strcpy(temp->dept, item.dept);
    strcpy(temp->desg, item.desg);
    temp->sal = item.sal;
    strcpy(temp->ph, item.ph);
    temp->llink = temp->rlink = NULL;
    if (first == NULL)
        return temp;
    cur = first;
    while (cur->rlink != NULL)
    {
        cur = cur->rlink;
    }
    cur->rlink = temp;
    temp->llink = cur;
    return first;
}
NODE delete_front(NODE first)
{
    NODE second;
    if (first == NULL)
    {
        printf("Employee list is empty\n");
        return NULL;
    }
    if (first->rlink == NULL)
    {
        printf("Employee details deleted: ssn=%d\n", first->ssn);
```

```
        free(first);
        return NULL;
    }
    second = first->rlink;
    second->llink = NULL;
    printf("Employee details: ssn=%d\n", first->ssn);
    free(first);
    return second;
}
NODE delete_rear(NODE first)
{
    NODE cur, prev;
    if (first == NULL)
    {
        printf("List is empty cannot delete\n");
        return first;
    }
    if (first->rlink == NULL)
    {
        printf("Employee details deleted: ssn=%d\n", first->ssn);
        free(first);
        return NULL;
    }
    prev = NULL;
    cur = first;
    while (cur->rlink != NULL)
    {
        prev = cur;
        cur = cur->rlink;
```

```
    }
    printf("Employee details deleted: ssn=%d\n", cur->ssn);
    free(cur);
    prev->rlink = NULL;
    return first;
}
void display(NODE first)
{
    NODE cur;
    int count = 0;
    if (first == NULL)
    {
        printf("Employee list is empty\n");
        return;
    }
    cur = first;
    while (cur != NULL)
    {
        printf("SSN-%d\nSALARY-%f\nNAME-%s\nDEPARTMENT-%s\nDESIGNATION-
%s\nPHONE-%s\n", cur->ssn, cur->sal, cur->name, cur->dept, cur->desg, cur->ph);
        cur = cur->rlink;
        count++;
    }
    printf("Number of employees=%d\n", count);
}
int main()
{
    NODE first;
    int choice;
    EMPL item;
```

```
   first = NULL;
   for (;;)
   {

printf("\n1.Insert_front\t2.Insert_rear\n3.Delete_front\t4.Delete_rear\n5.Display\t6.Exit\n
");
      printf("Enter your choice:\n");
      scanf("%d", &choice);
      switch (choice)
      {
         case 1:
               printf("SSN:");
               scanf("%d", &item.ssn);
               printf("NAME:");
               scanf("%s", &item.name);
               printf("DEPARTMENT:");
               scanf("%s", &item.dept);
               printf("SALARY:");
               scanf("%f", &item.sal);
               printf("DESIGNATION:");
               scanf("%s", &item.desg);
               printf("PHONE:");
               scanf("%s", &item.ph);
               first = insert_front(item, first);
               break;
         case 2:
               printf("SSN:");
               scanf("%d", &item.ssn);
               printf("NAME:");
               scanf("%s", &item.name);
```

```
                printf("DEPARTMENT:");
                scanf("%s", &item.dept);
                printf("SALARY:");
                scanf("%f", &item.sal);
                printf("DESIGNATION:");
                scanf("%s", &item.desg);
                printf("PHONE:");
                scanf("%s", &item.ph);
                first = insert_rear(item, first);
                break;
        case 3:
                first = delete_front(first);
                break;
        case 4:
                first = delete_rear(first);
                break;
        case 5:
                display(first);
                break;
        case 6:
                exit(0);
                break;
      }
   }
   return 0;
}
```

OUTPUT :

1.Insert_front 2.Insert_rear

3.Delete_front 4.Delete_rear

5.Display 6.Exit

Enter your choice:

1

SSN:1

NAME : XXX

DEPARTMENT : HR

SALARY : 100000

DESIGNATION : MANAGER

PHONE : 100

1.Insert_front 2.Insert_rear

3.Delete_front 4.Delete_rear

5.Display 6.Exit

Enter your choice :

1

SSN : 2

NAME : YYY

DEPARTMENT : HR

SALARY : 50000

DESIGNATION : ASSISTANT

PHONE : 200

1.Insert_front 2.Insert_rear

3.Delete_front 4.Delete_rear

5.Display 6.Exit

Enter your choice :

5

SSN - 2

SALARY - 50000.000000

NAME - YYY

DEPARTMENT - HR

DESIGNATION - ASSISTANT

PHONE - 200

SSN - 1

SALARY - 100000.000000

NAME - XXX

DEPARTMENT - HR

DESIGNATION - MANAGER

PHONE - 100

Number of employees=2

1.Insert_front 2.Insert_rear

3.Delete_front 4.Delete_rear

5.Display 6.Exit

Enter your choice :

2

SSN : 3

NAME : ZZZ

DEPARTMENT : PR

SALARY : 150000

DESIGNATION : SEN.MANAGER

PHONE : 234

1.Insert_front 2.Insert_rear

3.Delete_front 4.Delete_rear

5.Display 6.Exit

Enter your choice :

5

SSN - 2

SALARY - 50000.000000

NAME - YYY

DEPARTMENT - HR

DESIGNATION - ASSISTANT

PHONE - 200

SSN - 1

SALARY - 100000.000000

NAME - XXX

DEPARTMENT - HR

DESIGNATION - MANAGER

PHONE - 100

SSN - 3

SALARY - 150000.000000

NAME - ZZZ

DEPARTMENT - PR

DESIGNATION - SEN.MANAGER

PHONE - 234

Number of employees = 3

1.Insert_front 2.Insert_rear

3.Delete_front 4.Delete_rear

5.Display 6.Exit

Enter your choice :

3

Employee details: ssn = 2

1.Insert_front 2.Insert_rear

3.Delete_front 4.Delete_rear

5.Display 6.Exit

Enter your choice :

4

Employee details deleted: ssn = 3

1.Insert_front 2.Insert_rear

3.Delete_front 4.Delete_rear

5.Display 6.Exit

Enter your choice :

5

SSN - 1

SALARY - 100000.000000

NAME - XXX

DEPARTMENT - HR

DESIGNATION - MANAGER

PHONE - 100

Number of employees = 1

1.Insert_front 2.Insert_rear

3.Delete_front 4.Delete_rear

5.Display 6.Exit

Enter your choice :

6

9. Develop a Program in C for the following operations Singly Circular Linked Lists(SCLL) with the header nodes

a) **Represent and Evaluate a polynomial P(x,y,z) = 6x**
b) **Find the sum of the 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 operation.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define COMPARE(x,y) ((x==y)?0:(x>y)?1:-1)
struct node
{
   int coef;
   int xexp,yexp,zexp;
   struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
   NODE x;
   x=(NODE)malloc(sizeof(struct node));
   if(x==NULL)
   {
      printf("Running out of memory\n");
      exit(0);
   }
   return x;
}
NODE attach(int coef,int xexp,int yexp,int zexp,NODE head)
{
   NODE temp,cur;
```

```
    temp=getnode();
    temp->coef=coef;
    temp->xexp=xexp;
    temp->yexp=yexp;
    temp->zexp=zexp;
    cur=head->link;
    while(cur->link!=head)
    {
       cur=cur->link;
    }
    cur->link=temp;
    temp->link=head;
    return head;
}
NODE read_poly(NODE head)
{
    int i,j,coef,xexp,yexp,zexp,n;
    printf("Enter the no of terms in the polynomials:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
       printf("\nEnter the %d term:",i+1);
       printf("\nCoef=");
       scanf("%d",&coef);
       printf("\n\t\tEnter Pow(x) Pow(y) Pow(z):");
       scanf("%d",&xexp);
       scanf("%d",&yexp);
       scanf("%d",&zexp);
       head=attach(coef,xexp,yexp,zexp,head);
```

```
    }
    return head;
}
void display(NODE head)
{
    NODE temp;
    if(head->link==head)
    {
        printf("\npolynomial does not exist");
        return;
    }
    temp=head->link;
    while(temp!=head)
    {
        printf("%dx^%dy^%dz^%d",temp->coef,temp->xexp,temp->yexp,temp->zexp);
        temp=temp->link;
        if(temp!=head)
            printf("+");
    }
}
int poly_evaluate(NODE head)
{
    int x,y,z,sum=0;
    NODE poly;
    printf("\nEnter the value of x,y,z;\n");
    scanf("%d %d %d",&x,&y,&z);
    poly=head->link;
    while(poly!=head)
    {
```

```
        sum+=poly->coef*pow(x,poly->xexp)*pow(y,poly->yexp)*pow(z,poly->zexp);
        poly=poly->link;
    }
    return sum;
}
NODE poly_sum(NODE head1,NODE head2,NODE head3)
{
    NODE a,b;
    a=head1->link;
    b=head2->link;
    while(a!=head1 && b!=head2)
    {
        while(1)
        {
            if(a->xexp==b->xexp && a->yexp==b->yexp && a->zexp==b->zexp)
            {
                int coef=a->coef+b->coef;
                head3=attach(coef,a->xexp,a->yexp,a->zexp,head3);
                a=a->link;
                b=b->link;
                break;
            }
            if(a->xexp!=0||b->xexp!=0)
            {
                switch(COMPARE(a->xexp,b->xexp))
                {
                    case -1: head3=attach(b->coef,b->xexp,b->yexp,b->zexp,head3);
                             b=b->link;
                             break;
```

```
    case 0: if(a->yexp>b->yexp)
            {
               head3=attach(a->coef,a->xexp,a->yexp,a->zexp,head3);
               a=a->link;
               break;
            }
            else if(a->yexp<b->yexp)
            {
               head3=attach(b->coef,b->xexp,b->yexp,b->zexp,head3);
               b=b->link;
               break;
            }
            else if(a->zexp>b->zexp)
            {
               head3=attach(a->coef,a->xexp,a->yexp,a->zexp,head3);
               a=a->link;
               break;
            }
            else if(a->zexp<b->zexp)
            {
               head3=attach(b->coef,b->xexp,b->yexp,b->zexp,head3);
               b=b->link;
               break;
            }
    case 1: head3=attach(a->coef,a->xexp,a->yexp,a->zexp,head3);
            a=a->link;
            break;
}
break;
```

```
}
if(a->yexp!=0||b->yexp!=0)
{
   switch(COMPARE(a->yexp,b->yexp))
   {
      case -1: head3=attach(b->coef,b->xexp,b->yexp,b->zexp,head3);
               b=b->link;
               break;
      case 0: if(a->zexp>b->zexp)
            {
               head3=attach(a->coef,a->xexp,a->yexp,a->zexp,head3);
               a=a->link;
               break;
            }
            else if(a->zexp<b->zexp)
            {
               head3=attach(b->coef,b->xexp,b->yexp,b->zexp,head3);
               b=b->link;
               break;
            }
      case 1: head3=attach(a->coef,a->xexp,a->yexp,a->zexp,head3);
            a=a->link;
            break;
   }
   break;
}
if(a->zexp!=0||b->zexp!=0)
{
   switch(COMPARE(a->zexp,b->zexp))
```

```
            {
               case -1: head3=attach(b->coef,b->xexp,b->yexp,b->zexp,head3);
                        b=b->link;
                        break;
               case 1: head3=attach(a->coef,a->xexp,a->yexp,a->zexp,head3);
                        a=a->link;
                        break;
            }
            break;
         }
      }
   }
   while(a!=head1)
   {
      head3= attach(a->coef,a->xexp,a->yexp,a->zexp,head3);
      a=a->link;
   }
   while(b!=head2)
   {
      head3=attach(b->coef,b->xexp,b->yexp,b->zexp,head3);
      b=b->link;
   }
   return head3;
}
int main()
{
   NODE head,head1,head2,head3;
   int res,ch;
   head=getnode();
```

```
head1=getnode();
head2=getnode();
head3=getnode();
head->link=head;
head->link=head;
head1->link=head1;
head2->link=head2;
head3->link=head3;
while(1)
{
   printf("MENU\n");
   printf("\n1.Represent and evaluate a polynomial P(x,y,z)");
   printf("\n2.Find the sum of two polynomials POLY(x,y,z)");
   printf("\nEnter your choice:");
   scanf("%d",&ch);
   switch(ch)
   {
      case 1:printf("\nPolynomial evaluation P(x,y,z)\n");
             head=read_poly(head);
             printf("\nRepresentation of a polynomial:\n");
             display(head);
             res=poly_evaluate(head);
             printf("\nResult of Polynomial evaluation is:%d\n",res);
             break;
      case 2:printf("\nEnter POLY1(x,y,z):");
             head1=read_poly(head1);
             printf("\nEnter POLY2(x,y,z):");
             head2=read_poly(head2);
             printf("\nPolynomial 1 is\n");
```

```
            display(head1);
            printf("\nPolynomial 2 is\n");
            display(head2);
            printf("\nPolynomial addition result:\n");
            head3=poly_sum(head1,head2,head3);
            display(head3);
            break;
        case 3: exit(0);
      }
   }
   return 0;
}
```

OUTPUT:

MENU

1.Represent and evaluate a polynomial P(x,y,z)

2.Find the sum of two polynomials POLY(x,y,z)

Enter your choice:1

Polynomial evaluation P(x,y,z)

Enter the no of terms in the polynomials:5

Enter the 1 term:

Coef=6

Enter Pow(x) Pow(y) Pow(z):2 2 1

Enter the 2 term:

Coef=-4

Enter Pow(x) Pow(y) Pow(z):0 1 5

Enter the 3 term:

Coef=3

Enter Pow(x) Pow(y) Pow(z):3 1 1

Enter the 4 term:

Coef=2

Enter Pow(x) Pow(y) Pow(z):1 5 1

Enter the 5 term:

Coef=-2

Enter Pow(x) Pow(y) Pow(z):1 1 3

Representation of a 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,z;

1 1 1

Result of Polynomial evaluation is:5

MENU

1.Represent and evaluate a polynomial P(x,y,z)

2.Find the sum of two polynomials POLY(x,y,z)

Enter your choice:2

Enter POLY1(x,y,z):Enter the no of terms in the polynomials:5

Enter the 1 term:

Coef=6

Enter Pow(x) Pow(y) Pow(z):4 4 4

Enter the 2 term:

Coef=3

Enter Pow(x) Pow(y) Pow(z):4 3 1

Enter the 3 term:

Coef=5

Enter Pow(x) Pow(y) Pow(z):0 1 1

Enter the 4 term:

Coef=10

Enter Pow(x) Pow(y) Pow(z):0 1 0

Enter the 5 term:

Coef=5

Enter Pow(x) Pow(y) Pow(z):0 0 0

Enter POLY2(x,y,z):Enter the no of terms in the polynomials:5

Enter the 1 term:

Coef=8

Enter Pow(x) Pow(y) Pow(z):4 4 4

Enter the 2 term:

Coef=4

Enter Pow(x) Pow(y) Pow(z):4 2 1

Enter the 3 term:

Coef=30

Enter Pow(x) Pow(y) Pow(z):0 1 0

Enter the 4 term:

Coef=20

Enter Pow(x) Pow(y) Pow(z):0 0 1

Enter the 5 term:

Coef=3

Enter Pow(x) Pow(y) Pow(z):0 0 0

Polynomial 1 is

6x^4y^4z^4+3x^4y^3z^1+5x^0y^1z^1+10x^0y^1z^0+5x^0y^0z^0

Polynomial 2 is

8x^4y^4z^4+4x^4y^2z^1+30x^0y^1z^0+20x^0y^0z^1+3x^0y^0z^0

Polynomial addition result:

14x^4y^4z^4+3x^4y^3z^1+4x^4y^2z^1+5x^0y^1z^1+40x^0y^1z^0+20x^0y^0z^1+8x^0y^0z

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 , Postorder , Preorder**
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 info;
   struct node *llink;
   struct node *rlink;
};
typedef struct node *NODE;
NODE getnode()
{
   NODE x;
   x=(NODE)malloc(sizeof(struct node));
   if(x==NULL)
   {
      printf("out of memory\n");
      exit(0);
   }
   return x;
}
void preorder(NODE root)
```

```
{
   if(root!=NULL)
   {
      printf("%d\n",root->info);
      preorder(root->llink);
      preorder(root->rlink);
   }
}
void postorder(NODE root)
{
   if(root!=NULL)
   {
      postorder(root->llink);
      postorder(root->rlink);
      printf("%d\n",root->info);
   }
}
void inorder(NODE root)
{
   if(root!=NULL)
   {
      inorder(root->llink);
      printf("%d\n",root->info);
      inorder(root->rlink);
   }
}
```

```
NODE insert(int item,NODE root)
{
   NODE temp,cur,prev;
   temp=getnode();
   temp->info=item;
   temp->llink=NULL;
   temp->rlink=NULL;
   if(root==NULL)
      return temp;
   prev=NULL;
   cur=root;
   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;
   return root;
}
NODE search(int item,NODE root)
```

```
{
   if(root==NULL)
      return root;
   if(item==root->info)
      return root;
   if(item<root->info)
      return search(item,root->llink);
   return search(item,root->rlink);
}
void main()
{
   NODE root,cur;
   int choice,item;
   root=NULL;
   for(;;)
   {
      printf("1.Insert\t2.Preorder\t3.Postorder\t4.Inorder\t5.Search\n");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
               printf("Enter the item to be inserted\n");
               scanf("%d",&item);
               root=insert(item,root);
               break;
         case 2:
```

```
        if(root==NULL)
        {
           printf("Tree is empty\n");
           break;
        }
        printf("The given tree in tree form\n");
        printf("Preorder traversal is\n");
        preorder(root);
        printf("\n");
        break;
    case 3:
        if(root==NULL)
        {
           printf("Tree is empty\n");
           break;
        }
        printf("The given tree in tree form\n");
        printf("Postorder traversal is\n");
        postorder(root);
        printf("\n");
        break;
    case 4:
        if(root==NULL)
        {
           printf("Tree is empty\n");
           break;
```

```
            }
            printf("The given tree in tree form\n");
            printf("Inorder traversal is\n");
            inorder(root);
            printf("\n");
            break;
        case 5:
            printf("Enter the item to be searched:\n");
            scanf("%d",&item);
            cur=search(item,root);
            if(cur==NULL)
               printf("Item not found\n");
            else
               printf("Item found\n");
            break;
        default:exit(0);
      }
   }
}
```

OUTPUT:

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

1

Enter the item to be inserted

6

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

1

Enter the item to be inserted

9

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

1

Enter the item to be inserted

5

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

1

Enter the item to be inserted

2

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

1

Enter the item to be inserted

8

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

1

Enter the item to be inserted

15

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

1

Enter the item to be inserted

24

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

1

Enter the item to be inserted

14

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

1

Enter the item to be inserted

7

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

1

Enter the item to be inserted

8

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

1

Enter the item to be inserted

5

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

1

Enter the item to be inserted

2

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

2

The given tree in tree form

Preorder traversal is

6

5

2

2

5

9

8

7

8

15

14

24

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

3

The given tree in tree form

Postorder traversal is

2

2

5

5

7

8

8

14

24

15

9

6

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

4

The given tree in tree form

Inorder traversal is

2

2

5

5

6

7

8

8

9

14

15

24

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

5

Enter the item to be searched:

3

Item not found

1.Insert 2.Preorder 3.Postorder 4.Inorder 5.Search

5

Enter the item to be searched:

24

Item found

11.Develop a program in C for the following operations on Binary search Tree(BST) of integers.

a) Create a graph of n cities using Adjacency Matrix.

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

```
#include<stdio.h>
#include<stdlib.h>
int a[20][20],q[20],visited[20],reach[20],n,i,j,f = 0,r=-1;
int count = 0;
void bfs(int v)
{
   for (i = 1; i <= n; i++)
      if (a[v][i] && !visited[i])
         q[++r] = i;
   if (f<=r)
   {
      visited[q[f]] = 1;
      bfs(q[f++]);
   }
}
void dfs(int v)
{
   int i;
   reach[v] = 1;
   for (i = 1; i <= n; i++)
   {
      if (a[v][i] && !reach[i])
      {
         printf("\n%d->%d", v, i);
         count++;
         dfs(i);
      }
   }
}
int main()
{
   int v, choice;
```

```
printf("\nEnter the number of vertices:\n");
scanf("%d", &n);
for (i = 1; i <= n; i++)
{
   q[i] = 0;
   visited[i] = 0;
}
for (i = 1; i <= n - 1; i++)
   reach[i] = 0;
printf("Enter graph data in matrix form:\n");
for (i = 1; i <= n; i++)
   for (j = 1; j <= n; j++)
      scanf("%d", &a[i][j]);
printf("\n1.BFS\t2.DFS\t3.Exit\n");
printf("Enter your choice:\n");
scanf("%d", &choice);
switch (choice)
{
   case 1:
         printf("\nEnter the starting vertex:\n");
         scanf("%d", &v);
         bfs(v);
         if (v < 1 || v > n)
         {
            printf("\nBFS not possible\n");
         }
         else
         {
            printf("\nThe nodes which are reachable from %d:", v);
            for (i = 1; i <= n; i++)
            if (visited[i])
               printf("%d\t", i);
         }
         break;
   case 2:
         printf("\nEnter the starting vertex:\n");
         scanf("%d", &v);
         dfs(v);
```

```
            if (count == n - 1)
               printf("\nGraph is connected: ");
            else
               printf("\nGraph is not connected:");
            break;
      case 3:
            exit(0);
      default:
            printf("\nInvalid choice\n");
      }
      return 0;
}
```

OUTPUT :

```
Enter the number of vertices:
4
Enter graph data in matrix form:
0 1 1 0
0 0 1 1
0 0 0 1
0 0 0 0
1.BFS    2.DFS    3.Exit
Enter your choice:
1
Enter the starting vertex:
1
The nodes which are reachable from 1: 2    3    4
Enter the number of vertices:
4
Enter graph data in matrix form:
0 1 1 0
0 0 1 1
0 0 0 1
0 0 0 0
1.BFS    2.DFS    3.Exit
Enter your choice:
1
Enter the starting vertex:
```

2
The nodes which are reachable from 2: 3 4
Enter the number of vertices:
4
Enter graph data in matrix form:
0 1 1 0
0 0 1 1
0 0 0 1
0 0 0 0
1.BFS 2.DFS 3.Exit
Enter your choice:
1
Enter the starting vertex:
3
The nodes which are reachable from 3: 4
Enter the number of vertices:
4
Enter graph data in matrix form:
0 1 1 0
0 0 1 1
0 0 0 1
0 0 0 0
1.BFS 2.DFS 3.Exit
Enter your choice:
2
Enter the starting vertex:
1
1->2
2->3
3->4
Graph is connected:
Enter the number of vertices:
4
Enter graph data in matrix form:
0 1 1 0
0 0 1 1
0 0 0 1
0 0 0 0

1.BFS 2.DFS 3.Exit
Enter your choice:
2
Enter the starting vertex:
4
Graph is not connected:

12. Given File of N employee records with a set K of keys(4 – digits) 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. Develop a program in C that uses hash functions H:K->L as H(K)=K mod m (remainder method) and implement hashing technique to map a given key K to the addresses space L Resolve the collision (if any) using linear probing.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10
struct employee
{
   int id;
   char name[15];
};
typedef struct employee EMP;
EMP emp[MAX];
int a[MAX];
int create(int num)
{
   int key;
   key = num % 100;
   return key;
}
int getemp(EMP emp[], int key)
{
   printf("\nEnter emp id: ");
   scanf("%d", &emp[key].id);
   printf("\nEnter emp name: ");
   getchar();
   fgets(emp[key].name, sizeof(emp[key].name), stdin);
   return key;
}
void display()
{
   int i, ch;
```

```
    printf("\n1.Display ALL\n2.Filtered Display");
    printf("\nEnter the choice: ");
    scanf("%d", &ch);
    if(ch == 1)
    {
      printf("\nThe hash table is:\n");
      printf("\nHTKey\tEmpID\tEmpName");
      for(i = 0; i < MAX; i++)
        printf("\n\t%d\t\t%d\t\t%s", i, emp[i].id, emp[i].name);
    } else
    {
      printf("\nThe hash table is:\n");
      printf("\n\tHTKey\t\tEmpID\t\tEmpName");
      for(i = 0; i < MAX; i++)
        if(a[i] != -1)
        {
          printf("\n%d\t%d\t%s", i, emp[i].id, emp[i].name);
          continue;
        }
    }
}
void linear_prob(int key, int num)
{
    int flag, i, count = 0;
    flag = 0;
    if(a[key] == -1)
    {
      a[key] = getemp(emp, key);
    }
    else
    {
      a[key] = getemp(emp, key);
      printf("\nCollision Detected...!!!\n");
      i = 0;
      while(i < MAX)
      {
        if(a[i] != -1)
          count++;
```

```
            else
               i++;
            printf("\nCollision avoided successfully using LINEAR
PROBING\n");
            if(count == MAX)
            {
               printf("\n Hash table is full");
               display(emp);
               exit(1);
            }
            for(i = key; i < MAX; i++)
            if(a[i] == -1)
            {
               a[i] = num;
               flag = 1;
               break;
            }
            i = 0;
            while((i < key) && (flag == 0))
            {
               if(a[i] == -1)
               {
                  a[i] = num;
                  flag = 1;
                  break;
               }
               i++;
            }
         }
      }
}
int main()
{
   int num, key, i;
   int ans = 1;
   printf("\nCollision handling by linear probing: ");
   for(i = 0; i < MAX; i++)
   {
```

```
            a[i] = -1;
        }
        do
        {
            printf("\nEnter the data: ");
            scanf("%d", &num);
            key = create(num);
            linear_prob(key, num);
            printf("\nDo you wish to continue? (1/0): ");
            scanf("%d", &ans);
        }
        while(ans);
        display(emp);
        getchar();
        return 0;
}
```

OUTPUT :

Collision handling by linear probing:
Enter the data: 1
Enter emp id: 287
Enter emp name: XXX
Do you wish to continue? (1/0): 1

Enter the data: 4
Enter emp id: 562
Enter emp name: YYY
Do you wish to continue? (1/0): 1

Enter the data: 6
Enter emp id: 683
Enter emp name: ZZZ
Do you wish to continue? (1/0): 0

1.Display ALL
2.Filtered Display
Enter the choice: 1

The hash table is:

HTKey	EmpID	EmpName
0	0	
1	287	XXX
2	0	
3	0	
4	562	YYY
5	0	
6	683	ZZZ
7	0	
8	0	
9	0	

Enter the choice: 2
The hash table is:

HTKey	EmpID	EmpName
1	287	XXX
4	562	YYY
6	683	ZZZ