B G S INSTITUTE OF TECHNOLOGY

B G Nagar – 571448.

DEPARTMENT
OF
COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

Programme (UG/PG)	:	UG
Year / Semester	:	2nd Year/3rdsem
Course Code	:	17CSL38
Course Title	:	DATA STRUCTURE LABORATORY

Prepared By:

Mrs. RASHMI G R

Assistant Professor,

Department of CS&E

B G S Institute of Technology

B G S INSTITUTE OF TECHNOLOGY

VISION

BGSIT is commited to the cause of creating tomorrow's engineers by providing quality education inculcating ethical values.

MISSION

M1: Imparting quality technical education by nurturing a conducive learningenvironment.

M2: Offering professional training to meet industry requirements.

M3: Providing education with a moral-cultural base and spiritual touch.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION

To produce engineers by possessing good technical knowledge and ethics through quality education and research.

MISSION

M1: Achieve excellence by providing good infrastructure and competentfaculty.

M2: Strengthening the technical, soft skills, leadership qualities and ethicalvalues to meet the industry requirements.

M3: Facilitate experimental learning through research projects.

INSTRUCTIONS TO STUDENTS

Computer Lab Safety Rules for Protecting Equipment

Turn off the machine once you are done using it.

Do not plug in external devices without scanning them for computer viruses.

Try not to touch any of the circuit boards and power sockets when a device is connected to them and switched on.

Always maintain an extra copy of all your important data files.

General Safety Guidelines to be followed at all times

All users of the laboratory are to follow the directions of Academic/Laboratory Technician staff member.

Students should not attempt to repair, open, tamper or interfere with any of the computer, printing, cabling, air conditioning or other equipment in the laboratory.

Students should be aware of office ergonomic guidelines for correct posture when using computer equipment.

Please treat fellow users of the laboratory, and all equipment within the laboratory, with the appropriate level of care and respect.

DO's AND DON'TS

Do's

Enter the log register

Follow the dress code and ware ID card

Always keep quiet. Be considerate to other lab users.

Report any problems with the computer to the person in charge.

Shut down the computer properly and keep the chairs aligned before leaving the lab.

Know the location of the fire extinguisher and the first aid box and how to use them in case of an emergency.

Report any broken plugs or exposed electrical wires to your lecturer/laboratory technician immediately.

Don'ts

Do not eat or drink in the laboratory.

Do not use mobile phone.

Don't damage, remove, or disconnect any labels, parts, cables or equipment.

Avoid stepping on electrical wires or any other computer cables.

Do not install or download any software or modify or delete any system files on any lab computers.

If you leave the lab, do not leave your personal belongings unattended.

Do not open the system unit casing or monitor casing particularly when the power is turned on.

Do not insert metal objects such as clips, pins and needles into the computer casings. They may cause fire.

LABORATORY RUBRICS

1. **FOR 25 MARKS (2010 SCHEME)**

Sl. No.	Description	Marks
1	**Continuous Evaluation**	**15**
	a. Observation write up and punctuality	**2.5**
	b. Conduction of experiment and output	**5.0**
	c. Viva voice	**2.5**
	d. Record write up	**5.0**
2	**Internal Test**	**10**

2. **FOR 20 MARKS (2015 CBCS SCHEME)**

Sl. No.	Description	Marks
1	**Continuous Evaluation**	**12**
	a. Observation write up and punctuality	**2.0**
	b. Conduction of experiment and output	**4.0**
	c. Viva voice	**2.0**
	d. Record write up	**4.0**
2	**Internal Test**	**08**

3. **FOR 40 MARKS (2017 REVISED CBCS SCHEME)**

Sl. No.	Description	Marks
1	**Continuous Evaluation**	**30**
	a. Observation write up and punctuality	**5.0**
	b. Conduction of experiment and output	**10.0**
	c. Viva voice	**5.0**
	d. Record write up	**10.0**
2	**Internal Test**	**10**

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professionalengineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Ability to apply Mathematical Methodologies, Management Principles and Ethics, Electronics and Embedded Systems and Programming Technologies to solve real time problems.

PSO 2: Ability to apply software design and development practices to develop software in emerging areas such as Internet of Things, Data Management, Social Networking and Security, Cloud and High-Performance Computing.

COURSE OUTCOMES

Upon successful completion of this course, students should be able to:

CO1	Examine various types of data structure operations, sorting and searching operations.
CO2	Analyse the performance of stacks, queues, lists.
CO3	Implement all the applications of data structure using high level languages.
CO4	Design and apply appropriate data structure for solving problems.

CO-PO-PSO MAPPING

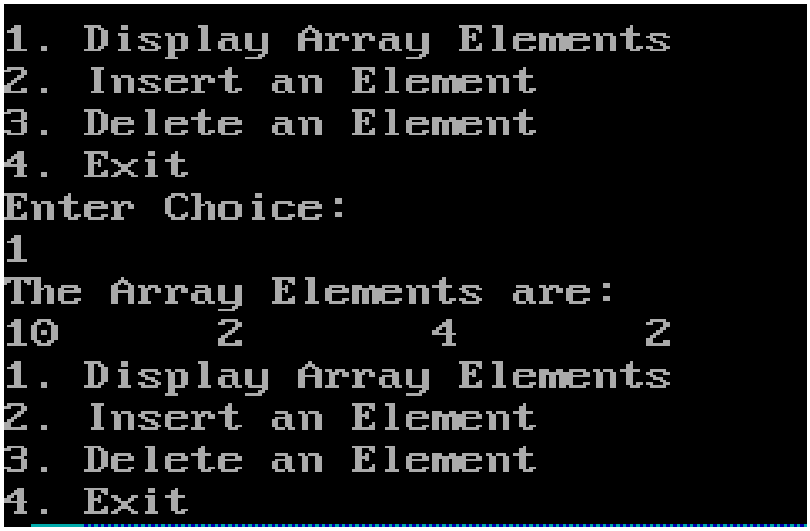
COs	**POs**												**PSOs**	
	PO1	PO 2	PO3	PO4	PO5	PO 6	P07	PO8	PO 9	P010	PO11	PO12	PSO1	PSO2
CO1	3	2	2	1	-	-	-	-	-	-	-	-	2	-
CO2	3	2	2	-	-	-	-	-	-	-	-	-	2	-
CO3	3	2	2	2	-	-	-	-	-	-	-	-	2	-
CO4	3	2	2	-	-	-	-	-	-	-	-	-	2	-
AVG	3	2	**2**	**1.5**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**2**	**-**

SubjectCode:18CSL38 **I.A. Marks :40**
Hours/Week:03 **Exam Hours:03**
Total Hours:50 **Exam Marks:80**

1. Design, Develop and Implement a menu driven Program in C for the followingArray operations
a. Creating an Array of N IntegerElements
b. Display of Array Elements with Suitable Headings
c. Inserting an Element (ELEM) at a given valid Position(POS)
d. Deleting an Element at a given valid Position(POS)
e. Exit.Support the program with functions for each of the aboveoperations.

```
#include<stdio.h>
#include <stdlib.h>
//Declaring Global Variables
int a[50],n,pos,element;
//Create an Array with n number of Elements
void createarray()
{
   int i;
   printf("Enter the NO OF ELEMENTS IN an array:");
   scanf("%d",&n);
   printf("enter array elements:\n");
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
}
//Display the Array Elements
void displayarray()
{
   int i;
   printf("The Array Elements are:\n");
   for(i=0;i<n;i++)
   printf("%d\t",a[i]);
}
//Insert an Element at the given Position
void insertelement()
{
   int i;
   printf("Enter the Element and Position to be Inserted\n");
   scanf("%d%d",&element,&pos);
   //Make space for the new element in the given position
   for(i=n-1;i>=pos;i--)
      a[i+1]=a[i];
   a[pos]=element;
   n++;          //Number of elements in the array afterinserting
}
//Delete an Element at the given Position
void deleteelement()
{
   int i;
   printf("Enter the Position of the Element to be Deleted\n");
   scanf("%d",&pos);
   printf("The Deleted Element is %d\n",a[pos]);
```

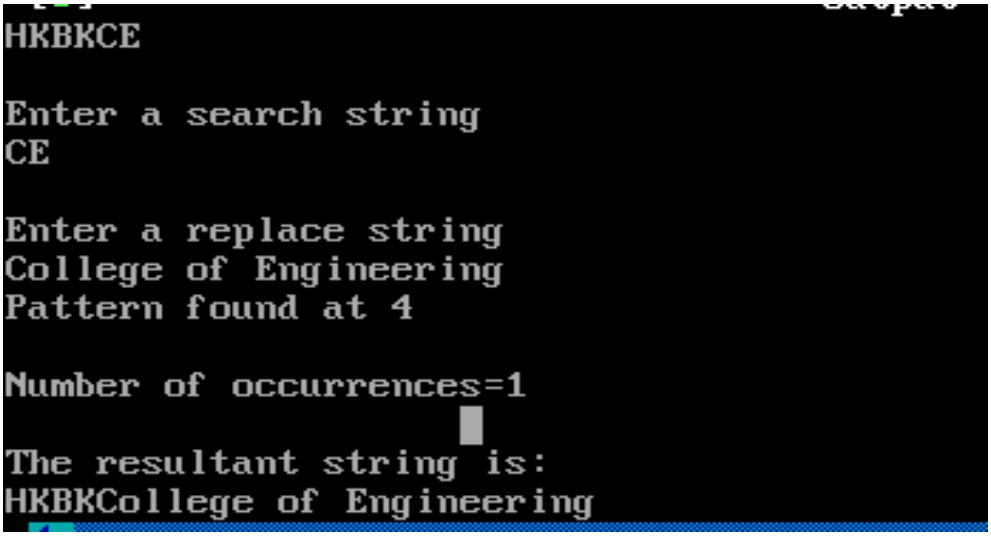
```
    //Delete by pushing up other elements

  for(i=pos;i<n;i++)
        a[i]=a[i+1];
    n--;          //Number of elements in the array afterdeleting
}
//main function
void main()
{
    int ch;
    createarray();
    while(1)
    {
        printf("\n1. Display Array Elements\n2. Insert an Element\n3. Delete an Element\n4. Exit\n");
        printf("Enter Choice:\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: displayarray();
                    break;
            case 2:insertelement();
                    break;
            case 3: deleteelement();
                    break;
            case 4: exit(0);
            default: printf("Invalid Choice\n");
        }
    }
}
```



2.Design, Develop and Implement a Program in C for the following operations onStrings
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 STRwith REP if PAT exists inSTR.
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>
void findandreplace(char str[100],char pat[100],char rep[100])
{
   int i,j,c,m,k,occ=0;
   char res[100];
   i=m=c=j=0;
         while(str[c]!='\0')
         {
                  if(str[m]==pat[i])     // CharacterMatched
                  {
                           i++;
                           m++;
         if(pat[i]=='\0')    //Pattern Found
         {
            occ++;
            printf("Pattern found at %d\n",c);
            for(k=0;rep[k]!='\0';k++,j++)    // Copy Replace String in resultString
               res[j]=rep[k];
            i=0;
            c=m;
         }
                  }
                  else                    //Pattern NotFound
                  {
                           res[j] = str[c];
                           j++;
                           c++;
                           m=c;
                           i=0;
                  }
         }   //while
         res[j]='\0';
         if (occ)
   {
      printf("\nNumber of occurrences=%d\n",occ);
      printf("\nThe resultant string is:\n%s\n" ,res);
   }
   else
      printf("Pattern not found\n");
}
```

```
void main()
{
        char str[100],pat[100],rep[100];

        printf("\nEnter a string \n");
        gets(str);
        printf("\nEnter a search string \n");
        gets(pat);
        printf("\nEnter a replace string \n");
        gets(rep);
        findandreplace(str,pat,rep);
}
```



3.Design, Develop and Implement a menu driven Program in C for the following operationson STACK of Integers(Array Implementation of Stack with maximum sizeMAX)
a. Push an Element on toStack
b. Pop an Element fromStack
c. Demonstrate how Stack can be used to checkPalindrome
d. Demonstrate Overflow and Underflow situations onStack
e. Display the status of Stack f. Exit Support the program with appropriate functions for each of the aboveoperations

```
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#define MAX 10
inttop=-1,a[MAX];
// function power
int power(int x,int n)
        {
                if(n==0) return 1;
                return (x*power(x,n-1));
        }
//Push Operation
void push(int item)
        {
                if(top==MAX-1)
                printf("Stack Overflow\n");
                else
                a[++top]=item;
        }
//Pop operation
 intpop()
        {
                int itemdel; if (top==-1) return 0; else
                {
                        itemdel=a[top--]; return itemdel;
                }
        }
//Display function
 void display()
        {
                        int i;
                        if(top==-1)
                        printf("Stack Empty\n");
                        else
                {
                        printf("Elements Are:\n"); for(i=top;i>=0;i--)
                        printf("%d\n",a[i]);
                }
```

```
//Palindrome
        void palindrome(int num)
                {
                        intcount=0,rem,i,rev=0,n=num,item; while(n!=0)
                                {
                                        rem=n%10; push(rem); n=n/10; count++;
                                }
                                        for(i=0;i<count;i++)
                                        {
                                                item=pop();
                                                 rev=item*power(10,i)+rev;
                                        }
                                                printf("Reversed Number=%d\n",rev); if(num==rev)
                                                printf("Palindrome"); else
                                                printf("Not a Palindrome");
                }
//Main function
 void main()
 {
        int ch,item,num,itemdel;
        while(1)
        {
                printf("\nEnter the Choice\n1.Push\n2.Pop\n3.Display\n4.Palindrome\n5.Exit\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: printf("Enter item to be inserted\n");
                                scanf("%d",&item);
                                push(item); break;
                        case 2: itemdel=pop();
                                if(itemdel)
                                printf("\n Deleted Item is:%d\n",itemdel);
                                else
                                printf("Stack Underflow\n");
                                break;
                        case 3: display();
                                break;
                        case 4: printf("Enter the Number:\n");
                                scanf("%d",&num);
                                palindrome(num);
                                break;
                        case 5: exit(0);
                }
        }
 }
```

```
Enter item to be inserted
20

Enter the Choice
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
3
Elements Are:
20
10
```

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 operators: +, -, *, /, %(Remainder),^(Power) and alphanumeric operands.

```
 #include<stdio.h>
 chara[25];
 inttop=-1;

void push(char symbol)
        {
                a[++top]=symbol;
        }

char pop()
        {
                Charitem;
                 item=a[top--];
                return(item);
        }

int precd(char op)
        {
                int r;
                 switch(op)
                {
                        case '^': r=3;break; case '*':
                        case '/':
                        case '%': r=2;break; case '+':
                        case '-':r=1;break;
                        case '(':r=0;break;
                        case '#': r=-1;break;
                }
                        return(r);
        }

        void infix_postfix(char infix[],char postfix[])
                {
                        inti,p=0;
                        charsymbol,item; push('#');
                                for (i=0;infix[i]!='\0';i++)
                                        {
                                                symbol=infix[i]; switch(symbol)
                                                        {
                                                                case '(': push(symbol); break;
                                                                case ')':item=pop();
                                                                while(item!='(')
                                                                {
                                                                        postfix[p++]=item; item=pop();
                                                                }
                                                        break; case '+':
                                                        case '-':
                                                        case '*':
```

```
                                        case '/':
                                        case '^':
                                        case '%': while(precd(a[top])>=precd(symbol))
                                {
                                        item=pop(); postfix[p++]=item;
                                }
                                        push(symbol); break;
                                        default: postfix[p++]=symbol; break;
                        }
                }
while(top>0)
                {
                        item=pop(); postfix[p++]=item;
                }
                        postfix[p]='\0';
        }

void main()
        {
                char infix[25],postfix[25];
                printf("Enter the Infix Expression:\n");
                scanf("%s",infix); infix_postfix(infix,postfix);
                printf("Postfix Expression: %s\n",postfix);
        }
```

5.Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^

```
#include<stdio.h>
#include<math.h>
float s[25];
inttop=-1;
float operation(char op,float op1,float op2)
{
   switch(op)
   {
      case '+': return(op1+op2);
      case '-': return(op1-op2);
      case '*': return(op1*op2);
      case '/': return(op1/op2);
      case '^': return(pow(op1,op2));
      case '%': return((int)op1%(int)op2);
   }
   return (0);
}
void push(float symbol)
{
   s[++top]=symbol;
}
float pop()
{
   return(s[top--]);
}
void main()
{
   char postfix[25],symbol;
   float op1,op2,res;int i;
   printf("Enter the Postfix Expression\n");
   scanf("%s",postfix);
   for(i=0;postfix[i]!='\0';i++)
   {
      symbol=postfix[i];
      if(isdigit(symbol))
         push(symbol-'0');
      else
      {
         op2=pop();
         op1=pop();
         res=operation(symbol,op1,op2);
         push(res);
      }
   }
   res=pop();
   printf("Result=%.2f",res);
}
```

```
Enter the Postfix Expression
12+
Result=3.00_
```

6. Design, Develop and Implement a menu driven Program in C for the following operationson Circular QUEUE of Characters (Array Implementation of Queue with maximum sizeMAX)
a. Insert an Element on to CircularQUEUE
b. Delete an Element from CircularQUEUE
c. Demonstrate Overflow and Underflow situations on CircularQUEUE
d. Display the status of CircularQUEUE
e. Exit
Support the program with appropriate functions for each of the above operations

```
#include <stdio.h>
#include<stdlib.h>
#define MAX 5
int a[MAX],f=0,r=-1,count=0;
void insert (int item)
{
   if (count==MAX)
      printf("Q Overflow\n");
   else
   {
      r=(r+1)%MAX;
      a[r]=item;
      count=count+1;
   }
}
int delet()
{
      int itemdel;
      if(count==0)
         return 0;
      else
      {
         itemdel=a[f];
         f=(f+1)%MAX;
         count=count-1;
         return(itemdel);
      }
}
void display()
{
      int i,j;
      if (count==0)
         printf("Qempty\n");
      else
      {
         i=f;
         for (j=1;j<=count;j++)
         {
            printf("%d\t",a[i]);
            i=(i+1)%MAX;
         }
      }
}
```

```
void main()
{
        int ch,item,itemdel;
        while (1)
        {
            printf("\nEnter the choice\n");
            printf("1.Insert\t2.Delete\t3.Display\t4.Exit\n");
            scanf("%d",&ch);
            switch(ch)
            {
                case 1: printf("Enter the item\n");
                        scanf("%d",&item);
                        insert(item);
                        break;
                case 2: itemdel=delet();
                        if (itemdel)
                            printf("The deleted item is %d\n",itemdel);
                        else
                            printf("Queue underflow\n");
                        break;
                case 3: display();
                        break;
                case 4: exit(0);
            }
    }
}
```

```
1.Insert        2.Delete        3.Display       4.Exit
1
Enter the item
10

Enter the choice
1.Insert        2.Delete        3.Display       4.Exit
3
10
Enter the choice
1.Insert        2.Delete        3.Display       4.Exit
4
```

7. Design, Develop and Implement a menu driven Program in C for the following operations onSingly Linked List (SLL) of Student Data with the fields: USN,Name, Branch, Sem, PhNo
a. Create a SLL of N Students Data by using frontinsertion.
b. Display the status of SLL and count the number of nodes init
c. Perform Insertion and Deletion at End ofSLL
d. Perform Insertion and Deletion at Front ofSLL
e. Demonstrate how this SLL can be used as STACK andQUEUE
f. Exit

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
structstudent
{
  char usn[12];
  char name[25];
  char branch[25];
  int sem;
  int phone_no;
  struct student*link;
};
typedef struct student STUD;

STUD *read_data()
{
    char usn[12],name[25],branch[25];
    int sem,phone_no;
    STUD *temp;
    temp=(STUD*)malloc(sizeof(STUD));
    printf("Enter the Students Details:\n");
    printf("Enter USN\n");
    scanf("%s",usn);
    strcpy(temp->usn,usn);
    printf("Enter Name\n");
    scanf("%s",name);
    strcpy(temp->name,name);
    printf("Enter Branch \n");
    scanf("%s",branch);
    strcpy(temp->branch,branch);
    printf("Enter Semester\n");
    scanf("%d",&sem);
    temp->sem=sem;
    printf("Enter Phone Number\n");
    scanf("%d",&phone_no);
    temp->phone_no=phone_no;
    temp->link=NULL;
    return temp;
}

STUD *insert_front(STUD *first)
{
      STUD *temp;
      temp=read_data();
```

```
    temp->link=first;
    return temp;
}

STUD *insert_end(STUD *first)
{
    STUD *temp,*prev;
    temp=read_data();
    if(first==NULL)
        return temp;
    prev=first;
    while(prev->link!=NULL)
        prev=prev->link;
    prev->link=temp;
    return first;
}

STUD *delete_front(STUD *first)
{
    STUD *cur;
    if(first==NULL)
    {
        printf("List is empty\n");
        return first;
    }
    cur=first;
    first=first->link;
    free(cur);
    return first;
}

STUD *delete_end(STUD *first)
{
    STUD *prev,*cur;
    if(first==NULL)
    {
        printf("List is empty\n");
        return first;
    }
    prev=NULL;
    cur=first;
    while(cur->link!=NULL)
    {
        prev=cur;
        cur=cur->link;
    }
    prev->link=NULL;
    free(cur);
    return first;
}
```

```
void display(STUD *first)
{
	STUD *temp;
	int count=0;
	if(first==NULL)
	{
	   printf("List is empty\n");
	   return;
	}
	printf("USN\tNAME\tBRANCH\tSEM\tPHONE NO.\n");
	temp=first;
	while (temp!=NULL)
	{
	   printf("%s\t%s\t%s\t%d\t%d\n",temp->usn,temp->name,temp->branch,temp->sem,temp->phone_no);
	   temp=temp->link;
	   count++;
	}
	printf("The number of nodes in SLL=%d\n",count);
}

void stack()
{
	int ch;
	STUD *first=NULL;
	while (1)
	{
	   printf("1.Push\n2.Pop\n3.Display\n4.Exit\n");
	   scanf("%d",&ch);
	   switch(ch)
	   {
	      case 1:first=insert_front(first);
		     break;
	      case 2:first=delete_front(first);
		     break;
	      case 3:display(first);
		     break;
	      case 4: return;
	   }
	}
}

void queue()
{
	int ch;
	STUD *first=NULL;
	while (1)
	{
	   printf("1.Insert\n2.Delete\n3.Display\n4.Exit\n");
	   scanf("%d",&ch);
	   switch(ch)
	   {
	      case 1: first=insert_end(first);
		     //display(first);
		     break;
```

```
            case 2: first=delete_front(first);
                    //display(first);
                    break;
            case 3:display(first);
                    break;
            case 4: return;
        }
    }
}

void main()
{
    int ch,i,n;
    STUD *first=NULL;
    printf("Creation of SLL of N Students\n");
    printf("Enter the number of students\n");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        first=insert_front(first);
    printf("SLL Created Successfully!!!\n");
    display(first);
    while(1)
    {
        printf("1.Display\n2.Insert End\n3:Delete End\n4.Insert Front\n5.Delete
        Front\n6.Stack\n7.Queue\n8.Exit\n");
        printf("Enter the choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:display(first);
                    break;
            case 2: first=insert_end(first);
                    printf("Node Inserted at theEnd\n");
                    break;
            case 3: first=delete_end(first);
                    printf("Node deleted at the End\n");
                    break;
            case 4: first=insert_front(first);
                    printf("Node Inserted at Front\n");
                    break;
            case 5: first=delete_front(first);
                    printf("Node deleted atFront\n");
                    break;
            case 6: stack();
                    break;
            case 7: queue();
                    break;
            case 8: exit(0);
        }
    }
}
```

```
1.Push
2.Pop
3.Display
4.Exit
3
USN        NAME       BRANCH    SEM
1          abc        cse       6
The number of nodes in SLL=1
1.Push
2.Pop
3.Display
4.Exit
```

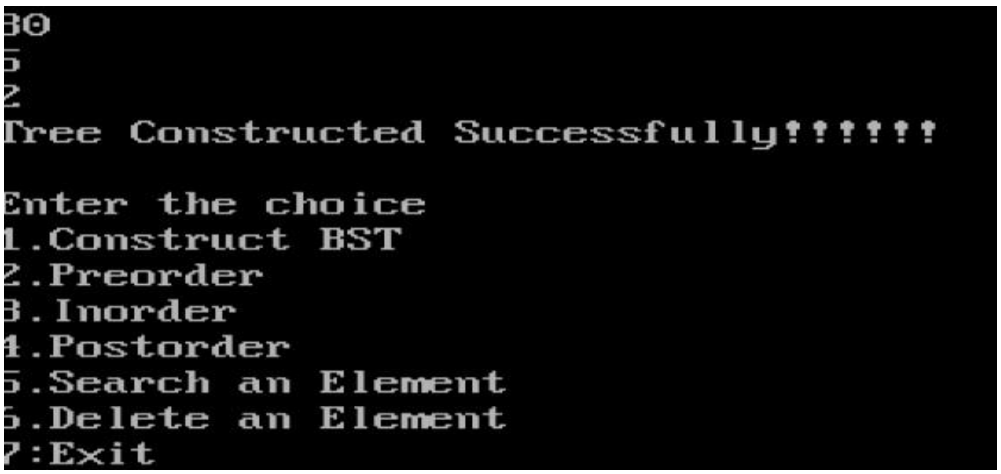
8. Design, Develop and Implement a menu driven Program inC for the following operations on Binary Search Tree (BST) of Integers
a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2
b. Traverse the BST in Inorder, Preorder and PostOrder
c. Search the BST for a given element (KEY) and report the appropriatemessage
d. Delete an element(ELEM) fromBST
e. Exit

```
#include<stdio.h>
#include<stdlib.h>
structnode
{
  int info;
  struct node *llink;
  struct node *rlink;
};
typedef struct node NODE;
NODE *insert(int item,NODE *root)
{
  NODE *temp,*cur,*prev;
  temp=(NODE*)malloc(sizeof(NODE));
  temp->info=item;
  temp->llink=NULL;
  temp->rlink=NULL;
  if(root==NULL)
  return temp;
  prev=NULL;
  cur=root;
  while(cur!=NULL)
  {
    prev=cur;
    cur=(item<=cur->info)?cur->llink:cur->rlink;
  }
  if(item<prev->info)
  prev->llink=temp;
  else
   prev->rlink=temp;
  return root;
}
NODE *construct_BST(NODE *root)
{
    int a,n,i;
    printf("Enter the number of elements\n");
    scanf("%d",&n);
    printf("Enter the elements to be inserted in the tree\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a);
        root=insert(a,root);
    }
    printf("Tree Constructed Successfully!!!!!!\n");
    return root;
    }
```

```
void preorder(NODE *root)
        {
          if(root!=NULL)
          {
            printf("%d\t",root->info);
            preorder(root->llink);
            preorder(root->rlink);
          }
        }
        void inorder(NODE *root)
        {
          if(root!=NULL)
          {
            inorder(root->llink);
            printf("%d\t",root->info);
            inorder(root->rlink);
          }
        }
        void postorder(NODE *root)
        {
          if(root!=NULL)
          {
            postorder(root->llink);
            postorder(root->rlink);
            printf("%d\t",root->info);
          }
        }
        int search_element(NODE *root,int key)
        {
            NODE *cur;
            int n=0;
            cur=root;
            if (cur!=NULL)
            {
                if (key==cur->info)
                    n=1;
                else if (key<cur->info)
                    return search_element(cur->llink,key);
                else
                    return search_element(cur->rlink,key);
            }
            else
                return n;
        }
        NODE *minValueNode(NODE* node)
        {
            NODE *current = node;
             /* loop to find the leftmost leaf */
            while (current->llink != NULL)
                current = current->llink;
            return current;
        }
```

```
NODE *delete_element(NODE *root,int key)
{
   if (root == NULL)
      return root;
   if (key < root->info)
      root->llink = delete_element(root->llink, key);
   else if (key > root->info)
      root->rlink = delete_element(root->rlink, key);
   else
   {
      // node with only one child or no child
      if (root->llink == NULL)
      {
         NODE *temp = root->rlink;
         free(root);
         return temp;
      }
      else if (root->rlink == NULL)
      {
         NODE *temp = root->llink;
         free(root);
         return temp;
      }
      // node with two children: Get the inorder successor (smallest
      // in the right subtree)
      else
      {
          NODE *temp = minValueNode(root->rlink);
          root->info = temp->info;
          root->rlink = delete_element(root->rlink, temp->info);
      }
   }
   return root;
}

void main()
{
  int item,ch,key,n;
  NODE *root;
  root=NULL;
  while (1)
  {
   printf("\nEnter the choice\n1.Construct BST\n2.Preorder\n3.Inorder\n4.Postorder\n5.Search an
Element\n6.Delete an Element\n7:Exit\n");
   scanf("%d",&ch);
   switch(ch)
   {
      case 1: root=construct_BST(root);
              break;
      case 2: preorder(root);
              break;
      case 3:inorder(root);
              break;
```

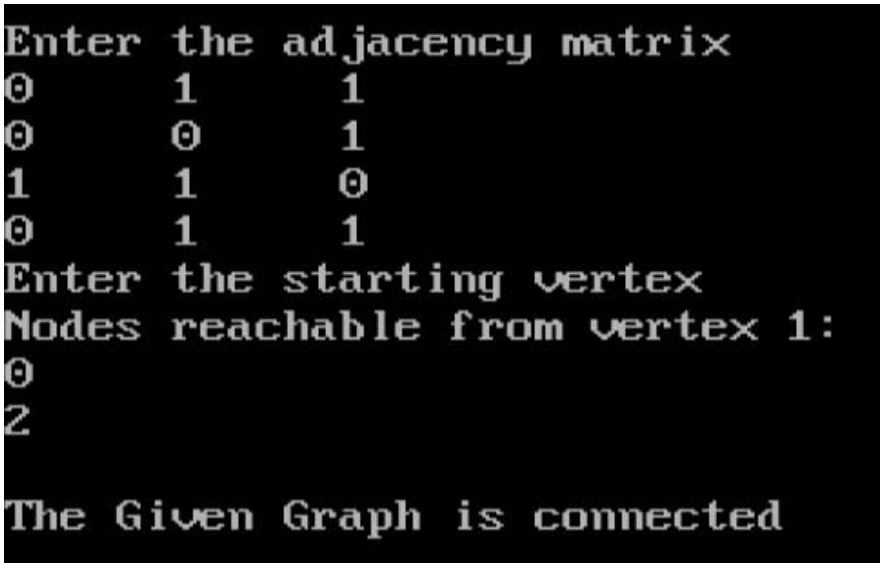
```
        case 4: postorder(root);
                break;
      case 5: if (root==NULL)
                printf("List Empty\n");
              else
              {
                printf("Enter the element\n");
                scanf("%d",&key);
                n=search_element(root,key);
                if(n)
                  printf("Key found\n");
                else
                  printf("Not found\n");
              }
              break;
      case 6: if (root==NULL)
                printf("List Empty\n");
              else
              {
                printf("Enter the element\n");
                scanf("%d",&key);
                n=search_element(root,key);
                if(n)
                  root=delete_element(root,key);
                else
                  printf("Not found\n");
              }
              break;
      case 7: exit(0);
      default: printf("Wrong Choice\n");
    }
  }
}
```



9. 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 BFSmethod
c. Check whether a given graph is connected or not using DFSmethod.

```
#include<stdio.h>

int a[10][10],q[10],visit[10],n,i,j;

void bfs(int v)
{
   static int f=0,r=-1;
          for(i=0;i<n;i++)
      if(a[v][i]==1 && visit[i]==0)
         q[++r]=i;
   if(f<=r)
   {
     visit[q[f]]=1;
     bfs(q[f++]);
   }
}

void dfs(int u)
{
   int i;
   visit[u]=1;
   for(i=0;i<n;i++)
      if(a[u][i]==1 && visit[i]==0)
         dfs(i);
}

main()
{
   int v,f=1;
          printf("\nEnter the number of nodes\n");
   scanf("%d",&n);
          printf("\nEnter the adjacency matrix\n");
          for(i=0;i<n;i++)
   for(j=0;j<n;j++)
          scanf("%d",&a[i][j]);
   for(i=0;i<n;i++)
      visit[i]=0;
   printf("Enter the starting vertex\n");
   scanf("%d",&v);
   visit[v]=1;
   bfs(v);

   printf("Nodes reachable from vertex %d:\n",v);
   for(i=0;i<n;i++)
          if(visit[i]==1 &&i!=v)
         printf("%d\n",i);
```

```
//Graph connected or not
        for(i=0;i<n;i++)
      visit[i]=0;
   dfs(0);
   for(i=0;i<n;i++)
      if(visit[i]==0)
      {
         f=0;
         break;
      }
   if(f==0)
      printf("\nThe Given Graph is not connected\n");
   else
      printf("\nThe Given Graph is connected\n");
}
```

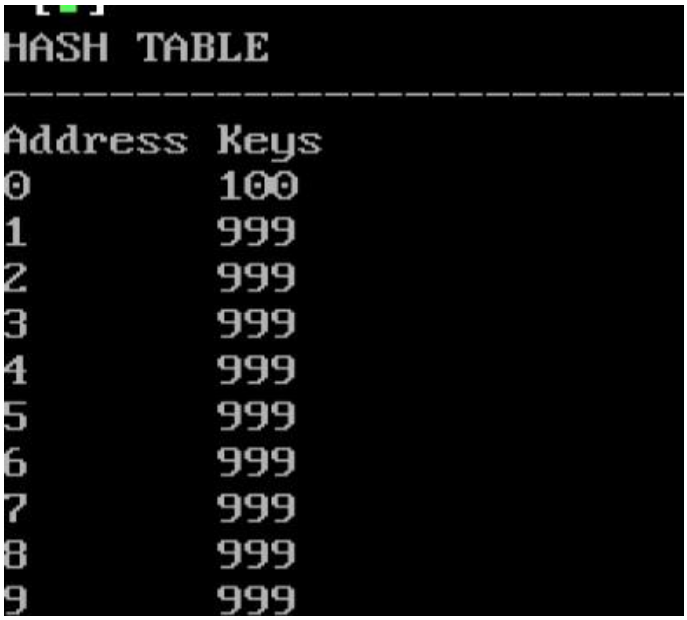


10. Given a File of N employee records with a set K of Keys(4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table(HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses 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 linearprobing.

```
#include <stdio.h>
#define m 10
int HT[10];
int hash(int key)
{
   return key%m;
}
void linear_probe(int hk,int key)
{
   int i,flag=0;
   for (i=hk+1;i<m;i++)
   {
      if (HT[i]==999)
      {
         HT[i]=key;
         flag=1;
         break;
      }
   }

   for(i=0;i<key&&flag==0;i++)
   {
      if (HT[i]==999)
      {
         HT[i]=key;
         flag=1;
         break;
      }
   }
   if (!flag)
      printf("HASH Table is Full!!!\n");
}
void main()
{
   int N,i,key,hk;
   for(i=0;i<m;i++)
      HT[i]=999;
   printf("Enter the Number of Employees:\n");
   scanf("%d",&N);
   printf("Enter the Employee Keys\n");
   for(i=1;i<=N;i++)
   {
      scanf("%d",&key);
      hk=hash(key);
      if (HT[hk]==999)
         HT[hk]=key;
```

```
      else
       {
          printf("Collision\n");
          //printf("Collision solved by Linear Probing\n");
          linear_probe(hk,key);
       }
    }
    printf("------------------------------\n");
    printf("HASH TABLE\n");
    printf("------------------------------\n");
    printf("Address\tKeys\n");
    for (i=0;i<m;i++)
       printf("%d\t%d\n",i,HT[i]);
}
```




```
/*solving collision using linear probing*/
#include <stdio.h>
#define m 20
int HT[10];
int hash(int key)
{
   return key%m;
}
void linear_probe(int hk,int key)
{
   int i,flag=0;
   for (i=hk+1;i<m;i++)
   {
      if (HT[i]==999)
      {
         HT[i]=key;
```

```
            flag=1;
            break;
        }
    }

    for(i=0;i<hk&&flag==0;i++)
    {
        if (HT[i]==999)
        {
            HT[i]=key;
            flag=1;
            break;
        }
    }
    if (!flag)
        printf("HASH Table is Full!!!\n");
}
void main()
{
    FILE *fp;
    int N,i,key,hk;
    char name[100];
    for (i=0;i<m;i++)
        HT[i]=999;
    fp=fopen("emp.txt","r");
    while(!feof(fp))
    {
        fscanf(fp,"%d%s",&key,name);
        hk=hash(key);
        if (HT[hk]==999)
            HT[hk]=key;
        else
        {
            printf("Collision for key %d:\n",key);
            printf("Collision solved by Linear Probing\n\n");
            linear_probe(hk,key);
        }
    }
    printf("-------------------------------\n");
    printf("HASH TABLE\n");
    printf("-------------------------------\n");
    printf("Address\tKeys\n");
    for (i=0;i<m;i++)
        printf("%d\t%d\n",i,HT[i]);
}
```

```
HASH Table is Full!!!
Collision for key 0:
Collision solved by Linear Probing
```