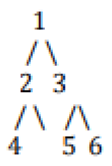
Laboratory Syllabus

Module-1
Laboratory Component:
1. Design, Develop and Implement a menu driven Program in C for the following Array Operations a. Creating an Array of N Integer Elements b. Display of Array Elements with Suitable Headings c. Exit. Support the program with functions for each of the above operations.
2. Design, Develop and Implement a menu driven Program in C for the following Array operations a. Inserting an Element (ELEM) at a given valid Position (POS) b. Deleting an Element at a given valid Position POS) c. Display of Array Elements d. Exit. Support the program with functions for each of the above operations.

Module-2
Laboratory Component:
1. Design, Develop and Implement 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 Overflow and Underflow situations on Stack
d. Display the status of Stack
e. Exit Support the program with appropriate functions for each of the above operations
2. Design, Develop and Implement a Program in C for the following Stack Applications
a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^
b. Solving Tower of Hanoi problem with n disks

Module-3
Laboratory Component:
1. Singly Linked List (SLL) of Integer Data
a. Create a SLL stack of N integer.
b. Display of SLL
c. Linear search. Create a SLL queue of N Students Data Concatenation of two SLL of integers.
2. Design, Develop and Implement a menu driven Program in C for the following operationson Doubly Linked List (DLL) of Professor Data with the fields: ID, Name, Branch, Area of specialization
a. Create a DLL stack of N Professor's Data.
b. Create a DLL queue of N Professor's Data Display the status of DLL and count the number of nodes in it.

Module-4
Laboratory Component:
1.Given an array of elements, construct a complete binary tree from this array in level order fashion. That is, elements from left in the array will be filled in the tree level wise starting from level 0.
Ex: Input : arr[] = {1, 2, 3, 4, 5, 6}
Output : Root of the following tree



2. Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers
a. Create a BST of N Integers
b. Traverse the BST in Inorder, Preorder and Post Order

Module-5
Laboratory Component:
1. 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 diagraph using DFS/BFS method.
2. Design and develop a program in C that uses Hash Function H:K->L as H(K)=K mod m(reminder method) and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing

1. Design, Develop and Implement a menu driven Program in C for the following Array Operations a. Creating an Array of N Integer Elements b. Display of Array Elements with Suitable Headings c. Exit. Support the program with functions for each of the above operations.

```
#include<stdio.h>
#include<stdlib.h>
int a[4], n, elem, i, pos;
void create()
{
printf("\nEnter the size of the array elements: ");
scanf("%d",&n);
printf("\nEnter the elements for the array:\n");
for(i=0;i<n;i++)
 scanf("%d",&a[i]);
}
void display()
{
int i;
printf("\nThe array elements are:\n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
}
void main()
{
 int ch;
 do
{
 printf("\n\n-------- Menu-----------\n");
 printf("1.Create\n 2.Display\n 3.Exit\n");
 printf("\nEnter your choice: ");
 scanf("%d",&ch);
 switch(ch)
 {
 case 1: create(); break;
case 2: display(); break;
case 3: exit(0); break;
default: printf("\nInvalidchoice:\n");
break;
}
```

OUTPUT:

```
-------- Menu-----------
1.Create
 2.Display
 3.Insert
 4.Delete
 5.Exit

Enter your choice: 1

Enter the size of the array elements: 3

Enter the elements for the array:
22
44
5


-------- Menu-----------
1.Create
 2.Display
 3.Insert
 4.Delete
 5.Exit

Enter your choice: 2

The array elements are:
22      44      5
```

2. Design, Develop and Implement a menu driven Program in C for the following Array operations a. Inserting an Element (ELEM) at a given valid Position (POS) b. Deleting an Element at a given valid Position POS) c. Display of Array Elements d. Exit. Support the program with functions for each of the above operations.

```c
#include<stdio.h>
#include<stdlib.h>
int a[4], n, elem, i, pos;
void insert()
{
printf("\nEnter the position for the new element:");
scanf("%d",&pos);
pos=pos-1;
printf("\nEnter the element to be inserted:"); scanf("%d", &elem);
for(i=n-1;i>=pos;i--)
{
a[i+1]=a[i];
}
a[pos]=elem;
n=n+1;
}
void del()
{
printf("\nEnter the position of the element to be deleted:");
scanf("%d",&pos);
pos=pos-1;
```

```
elem=a[pos];
for(i=pos;i<n-1;i++)
{
a[i]=a[i+1];
}
n=n-1;
printf("\nThe deleted element is = %d",elem);
}
void main()
{
 int ch;
 do
{
 printf("\n\n-------- Menu-----------\n");
 printf("1.Create\n 2.Display\n 3.Insert\n 4.Delete\n 5.Exit\n");
 printf("\nEnter your choice: ");
 scanf("%d",&ch);
 switch(ch)
 {
 case 1: create(); break;
case 2: display(); break;
case 3: insert(); break;
case 4: del(); break;
case 5: exit(0); break;
default: printf("\nInvalidchoice:\n");
break;
}
```

OUTPUT:

```
-------- Menu-----------
1.Create
 2.Display
 3.Insert
 4.Delete
 5.Exit

Enter your choice: 1

Enter the size of the array elements: 3

Enter the elements for the array:
22
44
5


-------- Menu-----------
1.Create
 2.Display
 3.Insert
 4.Delete
 5.Exit

Enter your choice: 2

The array elements are:
22       44       5
```

Module-2

1. Design, Develop and Implement 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 Overflow and Underflow situations on Stack

d. Display the status of Stack

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

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 4
int stack[MAX], item;
int ch, top=-1, count=0;

void push()
{
 if(top==(MAX-1))
 printf("\n\nStack is Overflow");
 else
 {
 top++;
 stack[top]=item;
 }
}
int pop()
 {
 int ret;
 if(top==-1)
 printf("\n\nStack is Underflow");
 else
 {
 ret=stack[top];
 top--;
 printf("\nPopped element is %d",ret);
 }
 return ret;
}
```

```
void palindrome()
{
int i, j,temp;
int flag=0;
for(i=0;j=top,i<j;i++,j--)
{
if(stack[i]!=stack[j])
{
flag=1;
break;
}
}
if(flag==0)
printf("\nStack contents are Palindrome");
else
printf("\nStack contents are not palindrome");
}
void display( )
{
int i;
printf("\nThe stack contents are:");
if(top==-1)
printf("\nStack is Empty");
else
{
for(i=top;i>=0;i--)
printf("\n ------\n| %d |", stack[i]);
printf("\n");
}
}
void main( )
{
do
{
printf("\n\n----MAIN MENU----\n");
printf("\n1. PUSH (Insert) in the Stack");
printf("\n2. POP (Delete) from the Stack");
printf("\n3. PALINDROME check using Stack");
printf("\n4. Exit (End the Execution)");
printf("\nEnter Your Choice: ");
scanf("%d",&ch);
switch(ch)
{
case 1: printf("\nEnter a element to be pushed: ");
scanf("%d",&item);
push();
display();
break;

case 2: pop();
display();
 break;
case 3:
palindrome();
break;
case 4:
exit(0); break;
 default:
printf("\nEND OF EXECUTION");
 }
}while(ch!=4);
}
```

Output:

```
---MAIN MENU----

1. PUSH (Insert) in the Stack
2. POP (Delete) from the Stack
3. PALINDROME check using Stack
4. Exit (End the Execution)
Enter Your Choice:1
Enter a element to be pushed: 2
The stack contents are:
 ------
| 2 |

----MAIN MENU----

1. PUSH (Insert) in the Stack
2. POP (Delete) from the Stack
3. PALINDROME check using Stack
4. Exit (End the Execution)
Enter Your Choice: 1

Enter a element to be pushed: 3

The stack contents are:
 ------
| 3 |
 ------
| 2 |

----MAIN MENU----

1. PUSH (Insert) in the Stack
2. POP (Delete) from the Stack
3. PALINDROME check using Stack
```

```
4. Exit (End the Execution)
Enter Your Choice:2
The poped element is 3
Stack is Overflow
The stack contents are:
 ------
| 5 |
 ------
| 4 |
 ------
| 3 |
 ------
| 2 |
```

2. Design, Develop and Implement a Program in C for the following Stack Applications
a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^
b. Solving Tower of Hanoi problem with n disks

```
#include<stdio.h>
#include<math.h>
#include<ctype.h>
#include<string.h>
double compute (char symbol, double op1, double op2)
{
switch(symbol)
{
case '+':
return op1 + op2;
case '-':
return op1 - op2;
case '*':
return op1 * op2;
case '/':
return op1 / op2;
case '$':
case '^':
return pow(op1,op2);
default:
return 0;
}
}
void main()
{
double s[20], res, op1, op2;
int top, i;
char postfix[20], symbol;
printf("\nEnter the postfix expression:\n");
scanf("%s",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:

```
To compile: cc ds5.c -lm
./a.out
Enter the postfix expression:
23+71-

The result is : 6.000000
```

2B.

```
#include<stdio.h>
#include<math.h>
void tower(int n, int source, int temp,int destination)
{
if(n == 0) return;
tower(n-1, source, destination, temp);
printf("\nMove disc %d from %c to %c", n, source, destination);
tower(n-1, temp, source, destination);
}
void main()
{
int n;
printf("\nEnter the number of discs: \n");
scanf("%d", &n);
tower(n, 'A', 'B', 'C');
printf("\n\nTotal Number of moves are: %d", (int)pow(2,n)-1);
}
```

OUTPUT:

```
To compile:    cc ds5B.c -lm
./a.out

Enter the number of discs:
2

Move disc 1 from A to B
Move disc 2 from A to C
Move disc 1 from B to C
```

Total Number of moves are: 3

Module-3
1. Singly Linked List (SLL) of Integer Data
a. Create a SLL stack of N integer.
b. Display of SLL
c. Linear search. Create a SLL queue of N Students Data Concatenation of two SLL of integers.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int MAX=4, count;
struct student
{
char usn[10];
char name[30];
char branch[5];
int sem;
char phno[10];
struct student *next;
};
typedef struct student NODE;
NODE *head;
int countnodes()
{
NODE *p;
count=0;
p=head;
while(p!=NULL)
{
p=p->next;
count++;
}
return count;
}

NODE *getnode()
{
NODE *newnode;
newnode=(NODE*)malloc(sizeof(NODE));
printf("\nEnter USN, Name, Branch, Sem, Ph.No\n");
scanf("%s",newnode->usn);
scanf("%s",newnode->name);
scanf("%s",newnode->branch);
scanf("%d",&(newnode->sem));
scanf("%s",newnode->phno);
newnode->next=NULL;
return newnode;
}
```

```
NODE *display()
{
NODE *p;
if(head==NULL)
printf("\nNo student data\n");
else
{
p=head;
printf("\n----STUDENT DATA----\n");
printf("\nUSN\tNAME\t\tBRANCH\tSEM\tPh.NO.");
while(p!=NULL)
{
printf("\n%s\t%s\t\t%s\t%d\t%s", p->usn, p->name, p->branch, p->sem, p->phno);
p=p->next;
}
printf("\nThe no. of nodes in list is: %d",countnodes(head));
}
return head;
}

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

void insert_front()
{
if(countnodes(head)==MAX)
printf("\nList is Full / Overflow!!");
else
head=create();
}

void insert_rear()
{
NODE *p, *newnode;
p=head;
```

```
if(countnodes(head)==MAX)
printf("\nList is Full(QUEUE)!!");
else
{
if(head==NULL)
{
newnode=getnode();
head=newnode;
}
else
{
newnode=getnode(head);
while(p->next!=NULL)
{
p=p->next;
}
p->next=newnode;
}
}
}

void insert( )
{
int ch;
do
{
printf("\n1.Insert at Front(First)\t2.Insert at End(Rear/Last)\t3.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)
{
case 1: insert_front(); break;
case 2: insert_rear(); break;
case 3: break;
}
display();
}while(ch!=3);
}

void delete_front()
{
NODE *p;
if(head==NULL)
printf("\nList is Empty/Underflow (STACK/QUEUE)");
else
{
p=head;
head=head->next;
free(p);
printf("\nFront(first)node is deleted");
}
}

void delete_rear()
{
NODE *p, *q;
p=head;
if(count==1)
{
delete_front();
return;
}
while(p->next!=NULL)
{
q=p;
p=p->next;
}
free(p);
q->next=NULL;
printf("\nLast(end) entry is deleted");
}

void del()
{
int ch;
do
{
printf("\n1.Delete from Front(First)\t2. Delete from End(Rear/Last))\t3.Exit");
printf("\nEnter your choice: ");
scanf("%d",&ch);
switch(ch)
{
case 1: delete_front();
break;
case 2: delete_rear();
break;
case 3: break;
}
display();
}while(ch!=3);
}

NODE *stack()
{
int ch;
do
{
printf("\nSSL used as Stack...");
printf("\n 1.Insert at Front(PUSH) \t 2.Delete from Front(POP))\t3.Exit");
printf("\nEnter your choice: ");
```

```
scanf("%d", &ch);
switch(ch)
{
case 1: insert_front();
break;
case 2: delete_front();
break;
case 3: break;
}
display();
}while(ch!=3);
}

void queue()
{
int ch;
do
{
printf("\nSSL used as Queue...");
printf("\n 1.Insert at Rear(INSERT) \t 2.Delete from Front(DELETE))\t3.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)
{
case 1: insert_rear();
 break;
case 2: delete_front();
break;
case 3: break;
}
display();
}while(ch!=3);
}

void main()
{
int ch, i, n;
head=NULL;
printf("\n*----------Studednt Database-----------*");
do
{
printf("\n 1.Create\t 2.Display\t 3.Insert\t 4.Delete\t 5.Stack\t 6.Queue\t 7.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)
{
case 1: printf("\nHow many student data you want to create: ");
scanf("%d",&n);
for(i=0;i<n;i++)
create();
```

```
break;
case 2: display();
break;
case 3: insert();
break;
case 4: del();
break;
case 5: stack();
break;
case 6: queue();
break;
case 7: exit(0);
}
```

OUTPUT:

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

How many student data you want to create: 2

Enter USN, Name, Branch, Sem, Ph.No
11
abc
cse
3
34556

Enter USN, Name, Branch, Sem, Ph.No
12
xyz
cse
4
67899

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

----STUDENT DATA----

USN     NAME            BRANCH          SEM     Ph.NO.
12      xyz             cse     4       67899
11      abc             cse     3       34556
The no. of nodes in list is: 2
 1.Create        2.Display       3.Insert        4.Delete        5.Stack         6.Queue
7.Exit
Enter your choice: 3
```

```
1.Insert at Front(First)2.Insert at End(Rear/Last)      3.Exit
Enter your choice: 1

Enter USN, Name, Branch, Sem, Ph.No
10
sow
aiml
3
86433

----STUDENT DATA----

USN     NAME            BRANCH      SEM    Ph.NO.
10      sow             aiml    3       86433
12      xyz             cse     4       67899
11      abc             cse     3       34556
The no. of nodes in list is: 3
1.Insert at Front(First)2.Insert at End(Rear/Last)      3.Exit
Enter your choice: 3

----STUDENT DATA----

USN     NAME            BRANCH      SEM    Ph.NO.
10      sow             aiml    3       86433
12      xyz             cse     4       67899
11      abc             cse     3       34556
The no. of nodes in list is: 3
 1.Create          2.Display        3.Insert          4.Delete          5.Stack           6.Queue
7.Exit
Enter your choice: 4

1.Delete from Front(First)     2. Delete from End(Rear/Last))         3.Exit
Enter your choice: 1

Front(first)node is deleted
----STUDENT DATA----

USN     NAME            BRANCH      SEM    Ph.NO.
12      xyz             cse     4       67899
11      abc             cse     3       34556
The no. of nodes in list is: 2
1.Delete from Front(First)     2. Delete from End(Rear/Last))         3.Exit
Enter your choice: 3

----STUDENT DATA----

USN     NAME            BRANCH      SEM    Ph.NO.
12      xyz             cse     4       67899
11      abc             cse     3       34556
The no. of nodes in list is: 2
```

```
 1.Create         2.Display       3.Insert          4.Delete          5.Stack            6.Queue
7.Exit
Enter your choice: 7
```

2. Design, Develop and Implement a menu driven Program in C for the following operationson Doubly Linked List (DLL) of Professor Data with the fields: ID, Name, Branch, Area of specialization
a. Create a DLL stack of N Professor's Data.
b. Create a DLL queue of N Professor's Data Display the status of DLL and count the number of nodes in it.

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int MAX=4, count;
struct emp
{
int ssn;
char name[20]; char dept[10]; char desig[15]; int sal;
char phno[10];
struct emp *left;
struct emp *right;
};
typedef struct emp NODE;
int countnodes(NODE *head)
{
NODE *p; count=0;
p=head;
while(p!=NULL)
{
p=p->right;
count++;
}
return count;
}
NODE* getnode(NODE *head)
{
NODE *newnode;
newnode=(NODE*)malloc(sizeof(NODE));
newnode->right=newnode->left=NULL;
printf("\nEnter SSN, Name, Dept, Designation, Sal, Ph.No\n");
scanf("%d",&newnode->ssn);
scanf("%s",newnode->name);
scanf("%s",newnode->dept);
scanf("%s",newnode->desig);
scanf("%d",&newnode->sal);
scanf("%s",newnode->phno);
head=newnode;
return head;
}
```

```
NODE* display(NODE *head)
{
NODE *p;
if(head==NULL)
printf("\nNo Employee data\n");
else
{
p=head;
printf("\n----EMPLOYEE DATA----\n");
printf("\nSSN\tNAME\tDEPT\tDESINGATION\tSAL\t\tPh.NO."); while(p!=NULL)
{
printf("\n%d\t%s\t%s\t%s\t\t%d\t\t%s", p->ssn, p->name, p->dept, p->desig,
p->sal, p->phno);
p = p->right;
}
printf("\nThe no. of nodes in list is: %d",countnodes(head));
}
return head;
}
NODE* create(NODE *head)
{
NODE *p, *newnode; p=head;
if(head==NULL)
{
newnode=getnode(head);
head=newnode;
}
else
{
newnode=getnode(head);
while(p->right!=NULL)
{
p=p->right;
}
p->right=newnode; newnode->left=p;
}
return head;
}
NODE* insert_end(NODE *head)
{
if(countnodes(head)==MAX)
printf("\nList is Full!!");
else
head=create(head);
return head;
}
NODE* insert_front(NODE *head)
{
NODE *p, *newnode;
if(countnodes(head)==MAX)
```

```
printf("\nList is Full!!");
else
{
if(head==NULL)
{
newnode=getnode(head);
head=newnode;
}
else
{
newnode=getnode(head);
newnode->right=head;
head->left=newnode;
head=newnode;
}
}
return head;
}
NODE* insert(NODE *head)
{
int ch; do
{
printf("\n 1.Insert at Front(First) \t 2.Insert at End(Rear/Last)\t3.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)
{
case 1: head=insert_front(head);
break;
case 2: head=insert_end(head);
break;
case 3:
break;
}
head=display(head);
}while(ch!=3);
return head;
}
NODE* delete_front(NODE *head)
{
NODE *p;
if(head==NULL)
printf("\nList is Empty (QUEUE)");
else
{
p=head;
head=head->right;
head->right->left=NULL;
free(p);
printf("\nFront(first)node is deleted");
```

```
}
return head;
}
NODE* delete_end(NODE *head)
{
NODE *p, *q; p=head;
while(p->right!=NULL)
{
p=p->right;
}
q=p->left;
q->right=NULL;
p->left=NULL;
free(p);
printf("\nLast(end) entry is deleted");
return head;
}
NODE *del(NODE *head)
{
int ch; do {
printf("\n1.Delete from Front(First)\t2. Delete from End(Rear/Last))\t3.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)
{
case 1: head=delete_front(head);
break;
case 2: head=delete_end(head);
break;
case 3: break;
}
head=display(head);
}while(ch!=3);
return head;
}
NODE* queue(NODE *head)
{
int ch, ch1, ch2;
do
{
printf("\nDLL used as Double Ended Queue");
printf("\n1.QUEUE- Insert at Rear & Delete from Front");
printf("\n2.QUEUE- Insert at Front & Delete from Rear");
printf("\n3.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)
{
case 1: do{
printf("\n1.Insert at Rear\t2.Delete from From Front\t3.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch1);
switch(ch1)
{
case 1: head=insert_end(head);
break;
case 2: head=delete_front(head);
break;
case 3: break;
}
}while(ch1!=3);
break;
case 2: do
{
printf("\n1.Insert at Front\t2.Delete from Rear\t3.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch2);
switch(ch2)
{
case 1:head=insert_front(head);
break;
case 2:head=delete_end(head);
break;
case 3: break;
}
}while(ch2!=3);
break;

case 3: break;
}
}while(ch!=3);
head=display(head);
return head;
}
void main()
{
int ch, i, n;
NODE *head;
head=NULL;
printf("\n----------Employee Database-----------");
do
{
printf("\n1.Create\t2.Display\t3.Insert\t4.Delete\t5.Queue\t6.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)
{
case 1: printf("\nHow many employees data you want to create: ");
scanf("%d",&n);
for(i=0;i<n;i++)
```

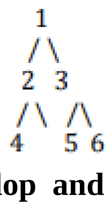
```
head=create(head);
break;
case 2: head=display(head);
break;
case 3: head=insert(head);
break;
case 4: head=del(head);
break;
case 5: head=queue(head);
break;
case 6: exit(0);
break;
}
}while(ch!=6);
}
```

OUTPUT:

```
----------Employee Database-----------
1.Create      2.Display     3.Insert      4.Delete      5.Queue       6.Exit
Enter your choice: 1

How many employees data you want to create: 2

Enter SSN, Name, Dept, Designation, Sal, Ph.No
123
sow
cse
ap
12000
23456

Enter SSN, Name, Dept, Designation, Sal, Ph.No
134
anu
aiml
ass.p
43000
321567

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

----EMPLOYEE DATA----

SSN   NAMEDEPT  DESINGATION      SAL           Ph.NO.
123   sow   cse   ap             12000         23456
134   anu   aiml  ass.p          43000         321567
The no. of nodes in list is: 2
1.Create      2.Display     3.Insert      4.Delete      5.Queue       6.Exit
Enter your choice: 3
```

```
 1.Insert at Front(First)           2.Insert at End(Rear/Last)     3.Exit
Enter your choice: 1

Enter SSN, Name, Dept, Designation, Sal, Ph.No
111
kavita
ise
ap
18000
78955

----EMPLOYEE DATA----

SSN     NAMEDEPT  DESINGATION        SAL             Ph.NO.
111     kavita  ise     ap                18000           78955
123     sow     cse     ap                12000           23456
134     anu     aiml    ass.p             43000           321567
The no. of nodes in list is: 3
 1.Insert at Front(First)           2.Insert at End(Rear/Last)     3.Exit
Enter your choice: 3

----EMPLOYEE DATA----

SSN     NAMEDEPT  DESINGATION        SAL             Ph.NO.
111     kavita  ise     ap                18000           78955
123     sow     cse     ap                12000           23456
134     anu     aiml    ass.p             43000           321567
The no. of nodes in list is: 3
1.Create        2.Display       3.Insert        4.Delete        5.Queue         6.Exit
Enter your choice: 6
```

Module-4
1.Given an array of elements, construct a complete binary tree from this array in level order fashion. That is, elements from left in the array will be filled in the tree level wise starting from level 0.
Ex: Input : arr[] = {1, 2, 3, 4, 5, 6}
Output : Root of the following tree



2. Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers
a. Create a BST of N Integers
b. Traverse the BST in Inorder, Preorder and Post Order

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

```
{
int data;
struct BST *left;
struct BST *right;
};
typedef struct BST NODE;
NODE *node;
NODE* createtree(NODE *node, int data)
{
if (node == NULL)
{
NODE *temp;
temp= (NODE*)malloc(sizeof(NODE));
temp->data = data;
temp->left = temp->right = NULL;
return temp;
}
if (data < (node->data))
{
node->left = createtree(node->left, data);
}
else if (data > node->data)
{
node -> right = createtree(node->right, data);
}
return node;
}
NODE* search(NODE *node, int data)
{
if(node == NULL)
printf("\nElement not found");
else if(data < node->data)
{
node->left=search(node->left, data);
}
else if(data > node->data)
{
node->right=search(node->right, data);
}
else
printf("\nElement found is: %d", node->data);
return node;
}
void inorder(NODE *node)
{
if(node != NULL)
{
inorder(node->left); printf("%d\t", node->data);
inorder(node->right);
}
```

```
}
void preorder(NODE *node)
{
if(node != NULL)
{
printf("%d\t", node->data); preorder(node->left);
preorder(node->right);
}
}
void postorder(NODE *node)
{
if(node != NULL)
{
postorder(node->left);
postorder(node->right);
printf("%d\t", node->data);
}
}
NODE* findMin(NODE *node)
{
if(node==NULL)
{
return NULL;
}
if(node->left)
return findMin(node->left);
else
return node;
}
void main()
{
int data, ch, i, n;
NODE *root=NULL;
while (1)
{
printf("\n1.Insertion in Binary Search Tree");
printf("\n2.Search Element in Binary Search Tree");
printf("\n4.Inorder\n5.Preorder\n6.Postorder\n7.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch (ch)
{
case 1: printf("\nEnter N value: " );
scanf("%d", &n);
printf("\nEnter the values to create BST like(6,9,5,2,8,15,24,14,7,8,5,2)\n");
for(i=0; i<n; i++)
{
scanf("%d", &data); root=createtree(root, data);
}
break;
case 2: printf("\nEnter the element to search: ");
scanf("%d", &data);
root=search(root, data);
break;
case 3: printf("\nInorder Traversal: \n");
inorder(root);
break;
case 4: printf("\nPreorder Traversal: \n"); preorder(root);
break;
case 5: printf("\nPostorder Traversal: \n");
postorder(root);
break;
case 7: exit(0);
default:printf("\nWrong option");
break;
}
}
}
```

OUTPUT:

```
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
4.Inorder
5.Preorder
6.Postorder
7.Exit
Enter your choice: 1

Enter N value: 5

Enter the values to create BST like(6,9,5,2,8,15,24,14,7,8,5,2)
3
2
6
7
1

1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
4.Inorder
5.Preorder
6.Postorder
7.Exit
Enter your choice: 4

Preorder Traversal:
3       2       1       6       7
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
4.Inorder
5.Preorder
```

```
6.Postorder
7.Exit
Enter your choice: 5

Postorder Traversal:
1	2	7	6	3
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
4.Inorder
5.Preorder
6.Postorder
7.Exit
Enter your choice: 6

Wrong option
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
4.Inorder
5.Preorder
6.Postorder
7.Exit
Enter your choice: 2

Enter the element to search: 2

Element found is: 2
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
4.Inorder
5.Preorder
6.Postorder
7.Exit
Enter your choice: 7
```

Module-5

1. 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 diagraph using DFS/BFS method.

```
#include<stdio.h>
#include<stdlib.h>
int a[10][10], n, m, i, j, source, s[10], b[10]; int visited[10];
void create()
{
printf("\nEnter the number of vertices of the digraph: ");
scanf("%d", &n);
printf("\nEnter the adjacency matrix of the graph:\n");
for(i=1; i<=n; i++)
for(j=1; j<=n; j++)
```

```
scanf("%d", &a[i][j]);
}
void bfs()
{
int q[10], u, front=0, rear=-1;
for(i=0;i<=n;i++)
visited[i]=0;
printf("\nEnter the source vertex to find other nodes reachable or not: ");
scanf("%d", &source);
q[++rear] = source;
visited[source] = 1;
printf("\nThe reachable vertices are: ");
while(front<=rear)
{
u = q[front++]; for(i=1; i<=n; i++)
{
if(a[u][i] == 1 && visited[i] == 0)
{
q[++rear] = i; visited[i] = 1; printf("\n%d", i);
}
}
}
}
void dfs(int source)
{
int v, top = -1; s[++top] = 1; b[source] = 1; for(v=1; v<=n; v++)
{
if(a[source][v] == 1 && b[v] == 0)
{
printf("\n%d -> %d", source, v); dfs(v);
}
}
}
void main()
{
int ch;
while(1)
{
printf("\n1.Create Graph\n2.BFS\n3.Check graph connected or not(DFS)\n4.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch); switch(ch)
{
case 1: create(); break; case 2: bfs();
for(i=1;i<=n;i++)
if(visited[i]==0)
printf("\nThe vertex that is not rechable %d" ,i);
break;
case 3: for(i=0;i<=n;i++)
b[i]=0;
printf("\nEnter the source vertex to find the connectivity: ");
scanf("%d",&source);
m=1;
dfs(source); for(i=1;i<=n;i++)
{
if(b[i]==0)
m=0;
}
if(m==1)
printf("\nGraph is Connected");
else
printf("\nGraph is not Connected");
break; default: exit(0);
}
}
}
```

OUTPUT:

```
1.Create Graph
2.BFS
3.Check graph connected or not(DFS)
4.Exit
Enter your choice: 1

Enter the number of vertices of the digraph: 4

Enter the adjacency matrix of the graph:
1 0 0 1
0 0 0 1
1 1 0 0
1 1 1 0

1.Create Graph
2.BFS
3.Check graph connected or not(DFS)
4.Exit
Enter your choice: 2

Enter the source vertex to find other nodes reachable or not: 1

The reachable vertices are:
4
2
3
1.Create Graph
2.BFS
3.Check graph connected or not(DFS)
4.Exit
Enter your choice: 3

Enter the source vertex to find the connectivity: 1
```

```
1 -> 4
4 -> 2
4 -> 3
Graph is Connected
1.Create Graph
2.BFS
3.Check graph connected or not(DFS)
4.Exit
Enter your choice: 4
```

2. Design and develop a program in C that uses Hash Function H:K->L as H(K)=K mod m(reminder method) and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing

```
#include <stdio.h>
#include <stdlib.h>
#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 % 10; return key;
}
int getemp(EMP emp[],int key)
{
printf("\nEnter emp id: ");
scanf("%d",&emp[key].id);
printf("\nEnter emp name: ");
scanf("%s",emp[key].name);
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%d\t%d\t%s", i, emp[i].id, emp[i].name);
}
else
{
printf("\nThe hash table is:\n"); printf("\nHTKey\tEmpID\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;
flag=0;
for(i=key+1;i<=MAX;i++)
{
if(a[i]==-1){
a[i]=getemp(emp,i);
flag=1;
break;
}
}
for(i=0;i<key&&flag==0;i++)
{
if(a[i]==-1)
{
a[i]=getemp(emp,i);
flag=1;
break;
}
}
if(!flag)
printf("hash tabl full:");
}
void main()
{
int num,key,i;
int ans=1;
printf("\nCollision handling by linear probing:");
for(i=0;i<MAX;i++){
a[i]=-1;
printf("%d",a[i]);
}
do{
printf("\nEnter the data:\n");
scanf("%d",&num);
key=create(num);
if(a[key]==-1)
a[key]=getemp(emp,key);
else
{
printf("collision detected\n");
linear_prob(key,num);
}
printf("\nDo you wish to continue?(1/0): ");
scanf("%d",&ans);
}while(ans);
display();
}
```

OUTPUT:

```
Enter the data:
2

Enter emp id: 122

Enter emp name: sow

Do you wish to continue?(1/0): 1

Enter the data:
4

Enter emp id: 121

Enter emp name: abc

Do you wish to continue?(1/0): 0

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

The hash table is:

HTKeyEmpIDEmpName
0       0
1       0
2       122     sow
3       0
4       121     abc
5       0
6       0
7       0
8       0
9       0
```