|| Jai Sri Gurudev ||

Sri Adichunchanagiri Shikshana Trust (R.)

BGS INSTITUTE OF TECHNOLOGY

[Affiliated to VTU, Belgaum; Approved by AICTE, New Delhi and Recognized by Govt. of Karnataka)

BG Nagara – 571 448 (Bellur Cross)
Nagamangala Taluk, Mandya District

Certificate

This is to certify that Mr/Ms. BHAVANA N.D

USN: 4BW17CS013 *has satisfactorily completed the course of experiments*

in Data structure *Laboratory (Course Code:* 17CSL38*)*

prescribed by the Visvesvaraya Technological University, Belagavi for .....

Semester, BE..... Computer Science ..... *Engineering,*

of this College in the year 2018 - 2019

Record Marks : 28 *Test Marks :* 06

IA Marks : 34

Date : 29/11/18

29/11/18
.....
Staff Incharge

Shashikala 29/11/18
.....
Head of the Department

INDEX

Name of the Student Bhavani.N.D ..... Class ..... Sem. 3rd

Expt. No	Date	Title of the Experiment	Page No.	Marks obtained					Sign. of the Staff
				a	b	c	d	T	
01	24/8/18	Array Operation	1-3	4	10	3	10	27	A
02	31/8/18	String Pattern Matching	4-6	4	10	4	10	28	A
03	31/8/18	Stack	7-10	4	10	4	10	28	A
04	7/9/18	Infix to postfix	11-14	4	10	4	10	28	A
04a	22/9/18	Suffix Expression	15-16	4	10	4	10	28	A
05	22/9/18	Towers of Hanoi	17-20	4	10	3	10	27	A
06	5/10/18	Circular Queue	21-23	4	10	3	10	27	A
07	12/10/18	Employee Detail	24-31	4	10	3	10	27	A
08	26/10/18	Student Detail	32-40	4	10	4	10	28	A
09	10/11/18	Polynomial	41-47	4	10	4	10	28	A
10	17/11/18	BST	48-54	4	10	4	10	28	A
11	17/11/18	Graph	55-56	4	10	4	10	28	A
12	17/11/18	Linear Probing	57-61	4	10	4	10	28	A

Name of Experiment ..... Date : .....

Experiment No. 01 Page No. 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. Inserting an Element at a given valid position.
d. Deleting an Element at a valid position.
e. Exit. Support the program with functions for each of the above equations.

```
# 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 (" 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 after
}
// 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 after
  }
// main function
  int main ()
{
   int ch;
   create array ();
   while (1)
{
Printf ("\n1. Display Array Elements \n2. Insert an Element \n3.
Delete an Element \n 4. Exit \n");
```

```
printf (" Enter choice :\n");
scanf (" %d", &ch);
switch (ch)
{
case 1 : display array()
         break;
case 2 : Insertelement()
         break;
case 3 : deleteelement()
         break;
case 4 : exit (0);
default : printf (" Invalid choice\n");
}
}
}
```

Seen
26/8/18

10/10

```
[root @ local host ~] # vi array123.c
[root @ local host ~] # cc array123.c
[root @ local host ~] # ./a.out

1] Enter the number of elements in an array
   6
enter array elements
 7   4   67   2   2   5
 1. Display array elements
 2. Insert an element
 3. Delete an element
 4. Exit
    Enter choice
    1
The array elements are
 7   4   67   2   2   5

 Enter the choice
  3
 Enter the position of the element to be deleted
   2   3
 Deleted element is 67
 1. Display array elements
 2. Insert an element
 3. Delete an element
 4. Exit
 Enter the choice
  1
```

The array elements are

7 4 2 2 5

1. Display array elements
2. Insert an element
3. Delete an element
4. Exit

Enter the choice

2

Enter the element and position to be Inserted

7 2

1. Display array elements
2. Insert an element
3. Delete an element
4. Exit

Enter the choice

1

The array elements are

7 4 7 2 2 5

1. Display array elements.
2. Insert an element.
3. Delete an element.
4. Exit

Design, Develop and Implement a program in C for the following operations on strings

a. Read a main string (STR), a pattern string (PAT) and a replace string
b. Perform pattern Matching Operation : Find and replace all occurences of PAT in STR with REP if PAT exists in STR.
Report suitable messages in case PAT does not exist in STR.
Support the program with functions for each of the above operations. Don't use Built-in functions.

```
#include <stdio.h>
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])  // Character method
    {
      i++;
      m++;
    if (pat[i]=='\0')  // pattern found
    {
      occ++;
printf("Pattern found at %d\n", c);
```

```
for (k=0; rep[k]!='\0'; k++, j++)
    res[j] = rep[k];
    e=0;
    c=m;
}
}
    else          // Pattern Not found
    {
    res[j] = str[c];
    j++;
    c++;
    m=c;
    e=0;
    }
}  // while
    res[j] = '\0';
    if (occ)
}
Printf("\n Number of occurences = %d\n", occ);
Printf("\n The resultant string is : \n%s\n", res);
}
  else
  Printf(" Pattern not found\n");
}
Int main()
{
  char str[100], pat[100], rep[100];
Printf("\n Enter a string\n");
```

```
    gets (str);
Printf ("\n Enter a search string \n");
    gets (pat);
Printf ("\n Enter a replace string \n");
    gets (rep);
find and replace (str, pat, rep);
}
```

Sem

10/10

```
[root @ local host ~] # vi pattern.c
[root @ local host ~] # cc pattern.c
[root @ local host ~] # ./a.out
```

1] Enter a string
Tumkur
Enter a search string
m
Enter a replace string
i
Pattern found at 1
number of occurances = 1
the resultant string is :
tuikur

2] Enter a string
Tumkur
Enter a search string
i
Enter a replace string
n
Pattern not found.

Name of Experiment .....

Experiment No. 03 Date : .....

Page No. 7

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 how stack can be used to check palindrome.
d. Demonstrate Overflow and underflow situations on stack
e. Display the status of stack f. Exit Support the program with appropriate functions for each of the above operations.

```
# include < stdlib.h>
# include < stdio.h>
# include < math.h>
# define MAX 10 int
top = -1, a [MAX]; // function power
int power (int x, int n)
{
  if (n == 0) return 1;
  return (x * power (x, n-1));
}
  if (top == MAX-1)
printf (" stack Overflow \n");
else
   a [++top] = item;
}
// Pop Operation
```

```
int pop()
{
 int item;
 if (top == -1)
  return 0;
 else
 {
  item = a[top--];
  return item;
 }
}
// 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)
{
 int count=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
int main()
{
    int ch, item, num, itemdel;
    while (1)
    {
Printf ("\n enter the choice \n 1. Push \n 2. Pop \n 3. Display
\n 4. Palindrome \n 5. Exit \n");
    scanf ("%d", &ch);
    switch (ch)
    {
```

```
case 1: Printf ("Enter item to be inserted\n");
        scanf ("%d", &item);
        push (item);
        breaks;
case 2: itemdel = pop();
        if (itemdel)
Printf ("\n Deleted Item is: %d\n", itemdel);
 else
Printf ("stack underflow\n");
 breaks;
case 3: display();
        breaks;
case 4: Printf ("Enter the Number:\n");
        scanf ("%d", &num);
        palindrome (num);
        breaks;
case 5: exit (0);
}
}
}
```

Scan

```
[root @ (local host in] # vi stacks.c
[root @ (local host in] # cc stacks.c
[root @ (local host in] # ./a.out
```

① Enter the choice
1. Push
2. Pop
3. display
4. Palindrome
5. exit
1

enter the item to be inserted
1

enter the choice
1. Push
2. Pop
3. Display
4. Palindrome
5. exit

enter the item to be inserted
2

enter the choice
1. Push
2. Pop
3. display
4. Palindrome
5. exit

enter the item to be inserted
3

enter the choice
1. Push
2. Pop
3. display
4. Palindrome
5. exit

enter the item to be inserted
4

enter the choice
1. Push
2. Pop
3. Display
4. Palindrome
5. exit

enter the item to be inserted
5

enter the choice
1. Push
2. Pop
3. display
4. Palindrome
5. exit

enter the item to be inserted

6

stack overflow

enter the choice

1. Push
2. Pop
3. display
4. Palindrome
5. exit

2

Item deleted is 5

enter the choice

1. Push
2. Pop
3. display
4. Palindrome
5. exit

2

stack underflow

enter the choice

1. Push
2. Pop
3. display
4. Palindrome
5. exit

3

stack empty

enter the choice
1. Push
2. Pop
3. display
4. Palindrome
5. exit
4
enter the number
121
reversed number 121 is palindrome

enter the choice
1. Push
2. Pop
3. display
4. Palindrome
5. exit
4
enter the number
1234

reversed number : 4321
not a palindrome

Enter the choice
1. Push
2. Pop
3. display
4. Palindrome
5. Exit

Date

Page No 11

Date of Experiment

Assignment No 04

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>
 char a[25];
 int top = -1;
void push (char symbol)
{
 a[++top] = symbol;
}
char pop()
{
 char item;
 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[])
{
    int i, p=0;
    char symbol, 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 [ptt] = item;
     }
          push (symbol);
          break;
   default:
        Postfix [ptt] = symbol;
        break;
   }
}
   while (top>0)
   {
      item = pop();
      postfix [ptt] = 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);
}
```

Seen
10/10

```
[root @ local host ~] # vi Infix.c
[root @ local host ~] # cc Infix.c
[root @ local host ~] # ./a.out
```

Output

(1) Enter the infix expression :
a+b
post fix expression : ab+

(2) Enter the infix expression :
a*b+c
post fix expression : ab*c+

Name of Experiment ..... Date : .....

Experiment No. ..... 05 [a] Page No. 15

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

```
#include <stdio.h>
#include <math.h>
float s[25];
int top=-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);
}
```

Seen 10/10

```
[root @ local host ~] # vi suffix.c
[root @ local host ~] # cc suffix.c
[root @ local host ~] # ./a.out
```

Output

(1) Enter the postfix expression
12+
result = 3.000.

(2) Enter the postfix expression
32+
result = 5.000

Name of Experiment ..... Date : .....

Experiment No. 05 [6] Page No. 17

Solving Tower of Hanoi problem with n disks

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <limits.h>
// Structure for
Stack struct stack
{
    int capacity;
    int top;
    int * array;
};
// Create a stack of given capacity
struct stack * create stack (int capacity)
{
    struct stack * stack = (struct stack *) malloc (size of
            (struct stack));
    stack -> capacity = capacity;
    stack -> top = -1; stack -> array = (int *)
        malloc (stack -> capacity * size of (int));
    return stack;
}
int is Full ( struct stack * stack)
{
    return (stack -> top == stack -> capacity - 1);
}
int is Empty (struct stack * stack)
{
```

```
    return (stack->top == -1);
}
void push (struct stack* stack, int item)
{
    if (is Full (stack))
    return;
  stack->array [++stack->top] = item;
}
int pop (struct stack * stack)
{
    if (is Empty (stack))
    return INT_MIN;
   return stack->array [stack->top--];
}
// Function to implement legal movement between
  // two poles
void move Disks Between Two Poles (struct stack * src, struct
  stack * dest, char s, char d)
{
  int pole1 Top Disk = pop (src);
  int pole2 Top Disk = pop (dest);
  // when pole 1 is empty
    if (Pole 1 Top Disk == INT_MIN)
  {
    push (src, pole2 Top Disk);
    move disk (d, s, pole2 Top Disk);
  }
  // when pole2 pole is empty
  else if (pole2 Top Disk == INT_MIN)
```

```
    {
        push (dest, pole1TopDisks);
        moveDisks (s, d, pole1TopDisks);
    }
// When top disk of pole1 > top disk of pole2
    else if (pole1TopDisks < pole2TopDisks)
    {
        push (src, pole1TopDisks);
        push (src, pole2TopDisks);
        moveDisks (d, s, pole2TopDisks);
    }
// When top disk of pole1 < top disk of pole2
    else
    {
        push (dest, pole2TopDisks);
        push (dest, pole1TopDisks);
        moveDisks (s, d, pole1TopDisks);
    }
}
// Function to show the movement of disks
void moveDisks (char fromPeg, char toPeg, int disks)
{
printf ("Move the disks from %d from \'%c\' to \'%c\'\n", disks, fromPeg, toPeg);
}
// Function to implement TOH puzzle
void tohIterative (int num of disks, struct stack *src, struct stack *aux, struct stack *dest)
{
```

```
    int i, total_num_of_moves;
    char s='S', d='D', a='A';
// If number of disks is even, then
  interchange // destination pole and auxiliary
  pole if (num of disks %2 ==0)
  {
    char temp = d;
    d=a; a=temp;
  }
  total num of moves = pow(2, num of disks)-1;
// Larger disks will be pushed first
  for (i=num of disks; i>=1; i--)
    push (src, i);
for (i=1; i<= total num of moves; i++)
  {
    if (i%3==1)
  move Disks Between Two poles (src, dest, s, d);
    else if (i%3==2)
  move Disks Between Two poles (src, aux, s, a);
    else if (i%3==0)
    move Disks Between Two poles (aux, dest, a, d);
  }
}
  int main()
{
  // Input : number of disks
  int num of disks;
Printf (" Enter the no of disks :");
```

```
    scanf("%d", &num_of_disks);
    struct stack *src, *dest, *aux;
// create three stacks of size "num_of_disks"
// to hold the disks
    src = createStack(num_of_disks);
    aux = createStack(num_of_disks);
    dest = createStack(num_of_disks);
tohIterative(num_of_disks, src, aux, dest);
    return 0;
}
```

26/10/18

10/16

```
[root @ local host ~] # vi tower.c
[root @ local host ~] # cc tower.c
[root @ local host ~] # ./a.out
```

Output

① Enter the number of disks : 3

move the disk 1 from 'S' to 'D'
move the disk 2 from 'S' to 'A'
move the disk 1 from 'D' to 'A'
move the disk 3 from 'S' to 'D'
move the disk 1 from 'A' to 'S'
move the disk 2 from 'A' to 'D'
move the disk 1 from 'S' to 'D'

② Enter the number of disks : 2

move the disk 1 from 'S' to 'A'
move the disk 2 from 'S' to 'D'
move the disk 1 from 'A' to 'D'

Name of Experiment ..... Date : .....

Experiment No. 06 Page No. 22

Design, Develop and Implement a menu driven program in C for the following operations on Circular QUEUE of Characters [Array Implementation of Queue with maximum size MAX].

a. Insert an Element on to Circular QUEUE.
b. Delete an Element from Circular QUEUE.
c. Demonstrate Overflow and underflow situations on Circular QUEUE.
d. Display the status of Circular QUEUE.
e. Exit.

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

```
# 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 item; if (count == 0)
  return 0;
 else
 {
  item = a[f];
  f = (f+1) % MAX;
  count = count - 1;
  return (item);
 }
}
void display()
{
  int i, j;
 if (count == 0)
printf ("\n empty\n");
 else
 {
   i = r;
for (j = 1; j <= count; j++)
 {
printf ("%d\t", a[i]);
   i = (i+1) % MAX;
 }
}
}
 void main()
{
```

```
    int ch, item, itemdel;
    while (1)
    {
    Printf ("\n Enter the choice\n");
    Printf ("1. Insert\t 2. Display\t 3. Delete\t 4. 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);
    }
  }
}
```

Seen

10/10

```
[root @ local host ~]: # cc Queue.c
[root @ local host ~]: # cc Queue.c
[root @ local host ~]: ./a.out
```

Output

```
1) Enter the choice.
   1. Insert  2. delete  3. display  4. exit.
   1
   enter Elem to be inserted
   10
2) Enter the choice
   1. Insert  2. delete  3. display  4. exit
   1
   Enter Elem to be inserted
   20
3) Enter the choice
   1. Insert  2. delete  3. display  4. exit
   1
   enter Elem to be inserted
   30
4) Enter the choice
   1. Insert  2. delete  3. display  4. exit
   40
5) Enter the choice
   1. Insert  2. delete  3. display  4. exit
   1
   enter Elem to be inserted
   50
```

8. Enter the choice
1. Insert 2. delete 3. display 4. exit
1
enter item to be inserted
60
① Overflow

9. Enter the choice
1. Insert 2. delete 3. display 4. exit
3
10 20 30 40 50

10. Enter the choice
1. Insert 2. delete 3. display 4. exit
2
the deleted item is 10.

11. Enter the choice
1. Insert 2. delete 3. display 4. exit
2
the deleted item is 30.

12. Enter the choice
1. Insert 2. delete 3. display 4. exit
2
The deleted item is 30.

13. Enter the choice
1. Insert 2. delete 3. display 4. exit
2

10% Enter the choice
1. Insert 2. delete 3. display 4. exit
2
The deleted item is 50.

13% Enter the choice
1. Insert 2. delete 3. display 4. exit
2
① underflow.

14% Enter the choice
1. Insert 2. delete 3. display 4. exit
4.

Name of Experiment ..... Date : .....

Experiment No. 7 Page No. 25

Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List of Student Data with the fields : USN, Name, Branch, 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 and Deletion at End of SLL.

d. Perform Insertion and Deletion at Front of SLL.

e. Demonstrate how this SLL can be used as STACK and QUEUE.

f. Exit.

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

```
}
snod * insert_end (snod * first)
{
  snod * temp, * prev;
  temp = read_data();
  if (first == NULL)
   return temp;
  prev = first;
  while (prev -> link != NULL)
   prev = prev -> link;
  prev -> link = temp;
   return first;
}
snod * delete_front (snod * first)
{
  snod * cur;
  if (first == NULL)
{
  printf ("list is empty\n");
  return first;
}
  cur = first;
  first = first -> link;
  free (cur);
  return first;
}
snod * delete_end (snod * first)
{
  snod *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\t BRANCH\t SEM\t PHONE 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;
  struct node *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;
    SLL * first = NULL;
    while (1)
{
printf ("1. Insert\n 2. Delete\n 3. Display\n 4. 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;
    SLL * first = NULL;
printf ("Creation of SLL of N students\n");
printf ("Enter the number of students\n");
scanf ("%d", &n);
for (i=0; i<=n; i++)
    first = Insert_front (first);
```

```
Printf ("SLL Created Successfully !!!\n");
display (first);
while (1)
{
Printf ("1. Display\n 2 Insert End\n 3. Delete End\n 4. Insert front
\n 5. Delete Front\n 6. Stack\n 7 Queue\n. 8. 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 the End\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 at front\n");
    break;
 Case 6 : stack ();
          break;
 Case 7 : queue ();
          break;
```

```
        case 8: exit (0);
    }
}
}
```

Sun
10/10

[root @ local host ~] :- #vi SLL.c
[root @ local host ~] :- cc SLL.c
[root @ local host ~] :- ./a.out

Output :-

```
Creation of SLL of N students
Enter the no of students
2
Enter the student details :-
Enter the USN
17CS059
Enter the name
Sanju
Enter the phone number
9147A8364 2
Enter the student details
Enter the USN
17CS009
Enter the Name
Bhavana
Enter the branch
CSE
Enter the semester
3rd
Enter the phone number
4139816 9
SLL Created Sucessfully !!!
```

USN	NAME	BRANCH	SEM	PHONE NO
17CS009	Bhavani	CSE	3	2147A83636.
17CS017	Sanju	CSE	3	2147A836A7.

the number of node in SLL = 2.

1. display
2. Insert end.
3. delete end.
4. Insert front
5. delete front
6. stack
7. queue
8. exit.

Enter the choice.

3

node deleted at end.

1. display
2. Insert end
3. delete end
4. Insert front
5. Delete front
6. stack
7. queue
8. exit.

Enter the choice

6

1. Push
2. Pop
3. Display
4. exit.

3

List is empty

1. Push
2. Pop
3. display
4. exit

1

Enter the student details

Enter the USN

17CS0A7

Enter the Name

Athyfa

Enter the Branch

ECE

Enter the semester

3

Enter the phone number

AC67689330

1. Push
2. Pop
3. display
4. exit

3

USN	NAME	BRANCH	SEM	Phone No
17CS0A7	Athyfa	ECE	3	AC67689330

the number of nodes in SLL = 1

1. Push
2. Pop
3. display
4. Exit.

3

1. Push
2. Pop
3. display
4. exit

3

List is empty

1. Push
2. Pop
3. display
4. exit

4

1. Display
2. Insert end
3. delete end
4. Insert front
5. delete front
6. stack
7. queue
8. exit.

Enter the choice.

7

1. Insert
2. delete
3. display
4. exit

3

List is empty

1. Insert
2. delete
3. display
4. exit.

1

Enter the student details

Enter the USN

17MEOA7

Enter the name

Praveen

Enter the Branch

Mech

Enter the semester:

5

Enter the phone number

1234567891

1. Insert
2. delete
3. display
4. exit

3

USN	NAME	BRANCH	SEM	PHONE NO
17MEOA7	Praveen	Mech	5	1234567891

the number of nodes in SLL = 1

1. Insert
2. delete
3. display
4. exit

2

1. Insert
2. delete
3. display
4. exit

3

List is empty

1. Insert
2. delete
3. display
4. exit
4
1. display
2. Insert end
3. delete end
4. Insert front
5. delete front
6. stack
7. Queue
8. Exit
Enter the choice
4
Enter the student details:
Enter the USN
17CS006
Enter the Name
Abhi
Enter the Branch
CSE
Enter the semester
3
Enter the phone number
1341567895
Node Insert at front.
1. display
2. Insert end
3. delete end

4. Insert front
5. delete front
6. stacks
7. queue
8. exit

Enter the choice.
1

USN	NAME	BRANCH	SEM	PHONE NO.
17CS006	ABHI	CSE	3	9147483647
17CS009	Bhavana	CSE	3	2341567890

the number of nodes in SLL = 2.

1. display
2. Insert end
3. delete end
4. Insert front
5. delete front
6. stacks
7. Queue.
8. Exit.

5

Node deleted at front

1. display
2. Insert end
3. delete end
4. Insert front
5. Delete front
6. stacks
7. Queue
8. exit

Enter the choice
1

USN	NAME	BRANCH	SEM	PHONE NO
17CS009	Bhavani	CSE	3	9147483647

the number of nodes in SLL :-1

1. display
2. Insert end
3. Delete end
4. Insert front
5. Delete front
6. stack
7. Queue
8. Exit.

8

Name of Experiment ..... Date : .....

Experiment No. 8 Page No. 33

Design, Develop and Implement a menu driven program in C for the following operations on Doubly Linked List of Employee Data with the fields: SSN, Name, Dept, Designation, Sal, PHNO

a. Create a DLL of N Employees data by using end insertion.

b. Display the status of DLL and count the number of nodes in it.

c. Perform insertion and deletion at End of DLL.

d. Perform insertion and deletion at front of DLL.

e. Demonstrate how this DLL can be used as Double Ended Queue.

f. Exit.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
  struct emp
  {
    char SSN [12];
    char name [25];
    char dept [25];
    char designation [25];
    float salary;
    int phone_no;
    struct emp *llink;
    struct emp *rlink;
  };
  typedef struct emp EMP;
```

```
EMP * read_data()
{
char SSN[12], name[25], dept[25], designation[25];
int phone_No;
float salary;
EMP *temp;
temp = (EMP *) malloc (size of (EMP));
Printf ("Enter the Employee details: \n");
Printf ("Enter SSN \n"); scanf ("%s", SSN);
strcpy (temp -> SSN, SSN);
Printf ("Enter Name \n");
scanf ("%s", name);
strcpy (temp -> name, name);
Printf ("Enter Department \n");
scanf ("%s", dept);
strcpy (temp -> dept, dept);
Printf ("Enter designation \n");
scanf ("%s", designation);
strcpy (temp -> designation, designation);
Printf ("Enter Salary \n");
scanf ("%f", & salary);
temp -> salary = salary;
Printf ("Enter Phone Number \n");
scanf ("%d", & phone_no);
temp -> phone_no = phone_no;
temp -> rlink = temp -> llink = NULL;
return temp;
}
EMP * insert_front (EMP *head)
```

```
}
 EHP *temp, *next;
 temp = read_data();
 next = head -> rlink;
 head -> rlink = temp;
 temp -> llink = head;
 next -> llink = temp;
 temp -> rlink = next;
 return head;
}
EHP *delete_end(EHP *head)
{
  EHP *cur, *prev;
if (head -> rlink == NULL)
printf("List is empty\n");
 else
{
 cur = head -> rlink;
 if (cur -> rlink == NULL)
head -> rlink = NULL;
 else
{
 while (cur -> rlink != NULL)
{
  prev = cur;
  cur = cur -> rlink;
}
prev -> rlink = NULL;
}
```

```
    free (cur);
}
return head;
}
EMP *delete_front (EMP *head)
{
  EMP *cur, *next;
if (head -> rlink == NULL)
printf ("List is empty\n");
else
{
cur = head -> rlink;
if (cur -> rlink == NULL)
head -> rlink = NULL;
else
{
 next = cur -> rlink;
next -> llink = head;
head -> rlink = next;
}
free (cur);
}
return head;
}
EMP *insert_end(EMP *head)
{
  EMP * cur, * temp;
temp = read_data();
cur = head;
```

```
while (cur->rlink != NULL)
 cur= cur->rlink;
 cur->rlink = temp;
 temp->llink = cur;
 return head;
}
EMP *deq (EMP *head)
{
  int ch;
  while (1)
 {
printf ("1. Insert front\n2. Insert end\n3. Delete Front\n4. Delete
End\n5. Display\n6. exit\n");
scanf ("%d", &ch);
 switch (ch)
{
 case 1 : head = insert_front (head);
      break;
 case 2 : head = insert_end (head);
      break;
 case 3 : head = delete_front (head);
      break;
 case 4 := head = delete_end (head);
      break;
 case 5 := display (head);
      break;
 case 6 ; return;
 }
}
```

```
}
void display (EMP *head)
{
  EMP *cur;
  int count=0;
 if (head->rlink==NULL)
 Printf("List is empty\n");
 else
 {
   cur=head->rlink;
Printf("SSN\t Name\t Dept\t Designation\t Salary\t Phone_No\n");
while (cur!=NULL)
{
Printf("%s\t %s\t %s\t %s\t %s\t %d\n", cur->ssn, cur->Name,
cur->dept, cur->designation, cur->salary, cur->phone_no);
  cur=cur->rlink;
  count++;
 }
Printf("Number of nodes in DLL=%d\n", count);
 }
}
void main()
{
 int ch,n,i;
 EMP *head;
 head=(EMP*)malloc(sizeof(EMP));
 head->rlink=head->llink=NULL;
 Printf("Creation of DLL\n");
 Printf("Enter the number of employees\n");
```

```
    scanf ("%d", &n);
for(i=1; i<=n; i++)
    head = insert_end(head);
    while(1)
{
printf("1. Display\n 2. Insert End\n 3. Delete End\n 4. Insert front
\n 5. Delete Front\n 6. Double Ended queue\n 7. Exit\n");
    printf(" Enter the choice\n");
    scanf("%d", &ch);
    switch(ch)
{
    case 1 : display(head);
            break;
    case 2 : head = insert_end(head);
            printf("Node inserted at the End\n");
            break;
    case 3 : head = delete_end(head);
            printf("Node deleted at the End\n");
            break;
    case 4 : head = insert_front(head);
            printf("Node Inserted at Front\n");
            break;
    case 5 : head = delete_front(head);
            printf("Node deleted at Front\n");
            break;
    case 6 : head = deq(head);
            break;
    case 7 : exit(0);
```

[root @ local host ~] :- # vi DLL.c .
[root @ local host ~] :- # cc dll.c .
[root @ local host ~] :- . la.out .

Output.

Creation of DLL

Enter the number of employees
2
Enter the employee details
Enter ssn
1
Enter the name
Dhanu
Enter the department
CSE
Enter the destination
lecture
Enter the salary
2000
Enter the phone number
1234
Enter the employee details:
Enter the SSN
2
Enter the name
Bhavani.
Enter the department
ECE

Enter the destination
Prof
Enter the Salary
3000
Enter phone number
12345
1. display
2. Insert end
3. Delete end
4. Insert front
5. Delete front
6. Double ended queue
7. Exit.
Enter the choice
1

SSN	Name	Dept	Designation	Salary	Phone no
1	Shaw	CSE	Lecturer	2000.00000	7259
2	Bhavani	CSE	teach	3000.00000	12345

Number of nodes in DLL = 2
1. display
2. Insert end
3. Delete end
4. Insert front
5. Delete front
6. Double ended queue
7. Exit
Enter the choice
2
Enter the employees details.
Enter the SSN
3

Enter the Name
Thanu
Enter the department
Mech
Enter the destination
Boring
Enter the Salary
1000
Enter phone number
3456
Node inserted at the end.
1. display
2. Insert end
3. delete end
4. Insert front
5. Delete front
6. Double ended queue
7. Exit
Enter the choice
1

SSN	Name	Dept	Destination	Salary	Phone-No
1	Dhanu	CSE	lecturer	2000.0000	1234
2	Bhavani	ECE	Teach	3000.0000	2345
3	Thanu	Mech	Boring	1000.0000	3456

Number of Nodes in DLL = 3
1. display
2. Insert end
3. Delete end
4. Insert front
5. Delete front

6. Search added queue
7. Exit

Enter the choice
5

Node deleted at the end

1. display
2. Insert end
3. Delete end
4. Insert front
5. Delete front
6. Search added queue
7. Exit

Enter the choice
1

car	Name	Dept	Destination	Salary	Phone-No
f	Shiva	CSE	Lecturer	30000	1234
i	Bhagavan	ECE	Head	30000	65545

Number of nodes in CLL = 2

1. display
2. Insert end
3. Delete end
4. Insert rear
5. Delete rear
6. Search added queue
7. Exit

Enter the choice
6

1. Insert front
2. Insert end
3. delete front
4. delete end

5. display
6. exit
1
Enter the employee's details
Enter SSN
c
Enter Name
d
Enter department
a
Enter designation
b
Enter the salary
23
Enter phone number
235A
1. Insert front
2. Insert end
3. delete front
4. delete end
5. display
6 exit
5

SSN	Name	dept	Designation	salary	Phone-No.
c	d	a	b	23.00000	235A
1	Dhanu	CSE	lecturer	2000.0000	123A
2	Bhavani	ECE	teach	3000.0000	2A5A5

Number of nodes in DLL : 3

1. Enert front
2. Enert end
3. delete front
4. delete end
5. display
6. Exit.

3

Enter the employee details:

Enter ssn

6

Enter the name

e

Enter departmen

8

Enter desegnation

9

Enter the salary

45

Enter the phone number

5677

1. Enert front
2. Enert end
3. delete front
4. delete end
5. display
6. exit

5

SSN	Name	dept	Designation	salary	Phone-No.
5	d	a	b	25.000000	0364
1	Dhana	CSE	lecture	2000.00000	0834
3	Bharani	ECE	Teach	3000.0000	03545

b c f g 45.00000 5677

Number of nodes in DLL : 4

1. Insert front
2. Insert end
3. delete front
4. delete end
5. display
6. exit

3

1. Insert front
2. Insert end
3. delete front
4. delete end
5. display
6. exit

5

SSN	Name	Dept	Designation	Salary	Phone-No
1	Ram	CSE	lecturer	2000.0000	1234
2	Hari	ECE	teach	3000.0000	23545
6	e	b	g	45.0000	35677

Number of nodes in DLL : 3

1. Insert front
2. Insert end
3. delete front
4. delete end
5. display
6. exit.

A

1. Insert front
2. Insert end
3. delete front
4. delete end
5. display
6. exit

5

SSN	Name	dept	designation	salary	Phone_no
1	Dhanu	CSE	lecture	2000.0000	1234
2	Bhavani	ECE	teach	3000.0000	23545

1. Insert front
2. Insert end
3. delete front
4. delete end
5. display
6. exit.

6.

2/11/19

10/10

Name of Experiment .....

Experiment No. 9 Date : .....

Page No. 41

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

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

b. Find the sum of two polynomials POLY1(x, y, z) and POLY2(x, y, z) and store the result in POLYSUM(x, y, z) Support the program with appropriate functions for each of the above operations.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct
{
  int coeff, x_exp, y_exp, z_exp;
  struct polynode *link;
} polynode;
polynode *createnode(int coeff, int x_exp, int y_exp, int z_exp)
{
   polynode *node;
  node = (polynode *) malloc(sizeof(polynode));
  node -> coeff = coeff;
  node -> x_exp = x_exp;
  node -> y_exp = y_exp;
  node -> z_exp = z_exp;
  node -> link = NULL;
   return node;
}
polynode *attachnode(polynode *node, polynode *poly)
```

```
{
    polynode *curr;
    curr = poly -> link;
    while (curr -> link != poly)
    {
        curr = curr -> link;
    }
    curr -> link = node;
    node -> link = poly;
    return poly;
}
polynode * readpoly()
{
    polynode *poly = (polynode *) malloc (sizeof(polynode));
    poly -> link = poly;
    int n, i;
    int coeff, x-exp, y-exp, z-exp;
    polynode *temp;
    printf ("Enter number of terms \n");
    scanf ("%d", &n);
    for (i=0; i<n; i++)
    {
     printf ("Term %d: \n", i+1);
     printf ("Enter the coefficient \n");
     scanf ("%d", &coeff);
     printf ("Enter exponent values for x, y and z \n");
     scanf ("%d %d %d", &x-exp, &y-exp, &z-exp);
     temp = createnode (coeff, x-exp, y-exp, z-exp);
     poly = attachnode (temp, poly);
```

```
    }
    return poly;
}
void display (polynode *poly)
{
    polynode *cur;
    cur = poly -> link;
    while (cur != poly)
    {
if (cur -> coeff >= 0)
printf ("+ %dx^%dy^%dz^%d", cur -> coeff, cur -> x-exp,
    cur -> y-exp, cur -> z-exp);
else
printf (" %dx^%dy^%dz^%d", cur -> coeff, cur -> x-exp, cur->y-
    exp, cur -> z-exp);
cur = cur -> link;
}
}
int power (int x, int n)
{
    if (n==0)
    return 1;
    return x * power (x, n-1);
}
void evaluate (polynode *poly)
{
    Polynode *poly;
    int x, y, z, xval, yval, zval, res=0;
    poly = poly -> link;
```

```
Printf ("\n Enter the values x,y and z:\n");
scanf ("%d%d%d", &x, &y, &z);
while (poly1 != poly2)
{
   xval = power (x, poly1->x_exp);
   yval = power (y, poly1->y_exp);
   zval = power (z, poly1->z_exp);
   res += poly1->coeff * xval * yval * zval;
   poly1 = poly1->link;
}
Printf ("Result = %d\n", res);
}
polynode *addpoly(polynode *poly1, polynode *poly2, polynode *poly)
{
   int comp;
   polynode *a, *b, *temp;
   a = poly1->link;
   b = poly2->link;
   while (a != poly1 && b != poly2)
   {
if (a->x_exp == b->x_exp && a->y_exp == b->y_exp && a->z_exp ==
      b->z_exp)
   comp = 0;
   else if (a->x_exp > b->x_exp)
    comp = 1;
   else
   if (a->x_exp == b->x_exp && a->y_exp > b->y_exp)
    comp = 1;
else if (a->x_exp == b->x_exp && a->y_exp == b->y_exp && a->z_exp > b
```

```
        b->z.exp)
      comp = 1;
      else
      comp = -1;
      switch (comp)
    {
    Case 0: temp = Create node (a->coeff+b->coeff, a->x.exp, a->y.exp,
            a->z.exp);
          poly = attachnode (temp, poly);
            a = a->link;
            b = b->link;
            break;
    Case 1: temp = Createnode (a->coeff, a->x.exp, a->y.exp, a->z.exp);
          poly = attachnode (temp, poly);
            a = a->link;
            break;
    Case-1: temp = Create node (b->coeff, b->x_exp, b->y.exp, b->z.exp);
          poly = attachnode (temp, poly);
            b = b->link;
            break;
    }
  }
    while (a != poly1)
  {
    temp = Createnode (a->coeff, a->x.exp, a->y.exp, a->z.exp);
    poly = attachnode (temp, poly);
    a = a->link;
  }
  while (b != poly2)
```

```
{
temp = Createnode (b->coeff, b->x-exp, b->y-exp, b->z-exp);
poly = attachnode (temp, poly);
b = b->link;
}
return poly;
}
void main ()
{
int ch;
polynode *poly1, *poly2, *poly3;
//poly1 = (polynode *) malloc (size of (polynode));
//poly2 = (polynode *) malloc (size of (polynode));
poly3 = (polynode *) malloc (size of (polynode));
//poly1 -> link = poly1;
//poly2 -> link = poly2;
poly3 -> link = poly3;
while (1)
{
printf ("\n\n 1. Represent & Evaluate a Polynomial\n2. Add 2
polynomial\n 3. Exit\n");
scanf ("%d", &ch);
switch (ch)
{
case 1 : printf ("Enter polynomial :\n");
poly1 = readpoly(); display (poly1);
evaluate (poly1);
break;
```

```
Case 2 : printf ("Enter polynomial 1:\n");
    poly1 = readpoly(); display (poly1);
printf ("\n Enter polynomial 2:\n");
poly2 = readpoly();
display (poly2);
poly3 = addpoly (poly1, poly2, poly3);
printf ("\n\n The Resultant Polynomial is:\n");
display (poly3);
break;
Case 3 : exit (0);
default : printf ("Wrong choice\n");
}
}
}
```

Sem

10/10

```
[root @ local host ~]: # vi sll.c
[root @ local host ~]: # cc sll.c
[root @ local host ~]: . la.out.
```

Output

```
1. represent and evaluate a polynomial
2. add 2 polynomial
3. exit
1
 enter polynomial :

 enter number of terms
 1
 term 1:
 enter the coefficients
 3
 enter the exponent values for x, y and z
  1  2  3
+3x^1y^2z^3
 enter the values x, y and z:
  2  3  2
 result : A32
```

```
1. represent and evaluate a polynomial
2. add 2 polynomial
3. exit

2
```

enter polynomial 1:

enter number of terms
1
term 1:
enter the coefficients
2
enter the exponent values for x, y and z
1 2 3
+2x^1y^2z^3

enter polynomial 2:

enter number of terms
1
term 1:
enter the coefficients
3
enter the exponent values for x, y and z
2 3 2
+3x^2y^3z^2

the resultant polynomial is
+3x^2y^3z^2 + 2x^1y^2z^3

3

Name of Experiment

Experiment No. 10 Date

Page No. 45

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 : 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 appropriate message
d. Delete an element (ELEM) from BST
e. Exit

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
  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", 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
```

```
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("\n Enter the choice\n1. Construct BST\n2. Preorder\n
3. 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("Tree 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("Tree 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");

}

}

}

Sum
16/11/19

10/10

[root @ localhost ~]: vi bst.c
[root @ localhost ~]: cc bst.c
[root @ localhost ~]: ./a.out

Output

enter the choice
1. Construct BST
2. Preorder
3. Postorder
4. Inorder
5. Search an element
6. Delete an element
7. exit
1
enter the number of elements
3
enter the element to be c
30
5
2
Tree constructed successfull !!!
enter the choice.
1. construct BST
2. Preorder
3. Postorder
4. Inorder
5. Search an element
6. Delete an element
7. exit.

2.

30 5 2
enter the choice
1. Construct BST
2. Pre-order
3. Post-order
4. In-order
5. Search an element
6. Delete an element
7. exit
3

2 5 30
enter the choice
1. Construct BST
2. Pre-order
3. Post-order
4. In-order
5. Search an element
6. Delete an element
7. exit.
4

2 30 5
enter the choice
1. Construct BST
2. Pre-order
3. Post-order
4. In-order
5. Search an element
6. Delete an element
7. exit

5

key found.

enter the choice.

1. Construct BST
2. Pre-order
3. Post-order
4. In-order
5. Search an element
6. Delete an element
7. exit.

5

enter the element

1

not found

enter the choice

1. Construct BST
2. Pre-order
3. Post-order
4. In-order
5. Search an element
6. Delete an element
7. exit

7

Name of Experiment ..... Date .....

Experiment No 11 Page No 28

Design, Develop and Implement a program in C for the following operations on Graph

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 BFS method

c. Check whether a given graph is connected or not using DFS method

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

```
main()
{
  int v, i=1;
printf("\n Enter the number of nodes\n");
scanf("%d", &n);
printf("\n Enter 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 ("\n The Given Graph is not connected\n");
else
Printf ("\n The Graph is connected\n");
}
```

Sun
10
10

```
[root @ local host ~]:- vi graph.c
[root @ local host ~]:- cc graph.c
[root @ local host ~]:- ./a.out
```

Output :-

```
enter the number of nodes
3

enter the adjacency matrix
1 1 1
1 1 1
1 1 1
enter the starting vertex
1
nodes reachable from vertex 1
0
2
The given graph is connected

enter the no of nodes
3
enter the adjacency matrix
0 1 0
1 1 0
0 0 1
```

enter the number of employee

12

enter the employee key

1

2

3

4

5

6

7

8

9

13

Collision

Hash table is full !!!

Hash Table

Index	Key
0	[illegible]
1	[illegible]
2	[illegible]
3	[illegible]
4	[illegible]
5	[illegible]
6	[illegible]
7	[illegible]
8	[illegible]
9	[illegible]

Name of Experiment ..... Date : .....

Experiment No. ..... Page No. 58

Given a file of N employee records with a set K of key & digit which uniquely determine the records in file F. Assume that file F is maintained in memory in a hash table of a memory locations with L as the set of memory addresses of locations in hash table. 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) mod m and implement hashing technique to map a given key K to the address space L. Resolve the collision using linear probing.

```
#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<size && flag==0; i++)
  {
    if (ht[i][i]==999)
    {
      ht[i][i] key;
      flag=1;
      break;
    }
  }
  if (flag)
  printf("HASH TABLE IS FULL!!!\n");
}
int main()
{
  int n, i, key, hs;
  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++)
  { Sem 28/11/18
    scanf("%d", &key);
    hs = hash(key);
    if (ht[hs]==999)
      ht[hs] = key;
```

10/10

[root @ local host] ~ :: vi out.txt
[root @ local host ~] :: cc out.txt
[root @ local host ~] :: ./a.out

Hash table

address	keys
0	999
1	999
2	999
3	123
4	999
5	999
6	999
7	999
8	999
9	999