



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(Accredited by NBA)**



**DATA STRUCTURES LABORATORY -
BCSL305**

PREPARED BY:

Prof. Sridevi N
Assistant Professor
Dept of CSE SVCE, Bangalore

Prof. Lokesh M
Assistant Professor
Dept of CSE SVCE, Bangalore

Prof. Archana M
Assistant Professor
Dept of CSE SVCE, Bangalore

VISION

To be a school of Excellence in Computing for Holistic Education and Research

MISSION

M1: Strive for academic excellence in Computer Science and Engineering through student centric innovative teaching learning process, competent faculty members, efficient assessment and effective use of ICT.

M2: Establish Centre for Excellence in various vertical of Computer Science and Engineering to promote collaborative research and Industry Institute Interaction.

M3: Transform the engineering aspirants to socially responsible, ethical, technically competent and value added professional or entrepreneur.

PROGRAMME OUTCOMES

- PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- PO2. 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.
- PO3. 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.
- PO4. 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.
- PO5. 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.
- PO6. 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.
- PO7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- PO9. Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- PO10. 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.

PO11. 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.

PO12. 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 adopt quickly for any domain, interact with diverse group of individuals and be an entrepreneur in a societal and global setting.

PSO 2. Ability to visualize the operations of existing and future software Applications.

Course Objectives

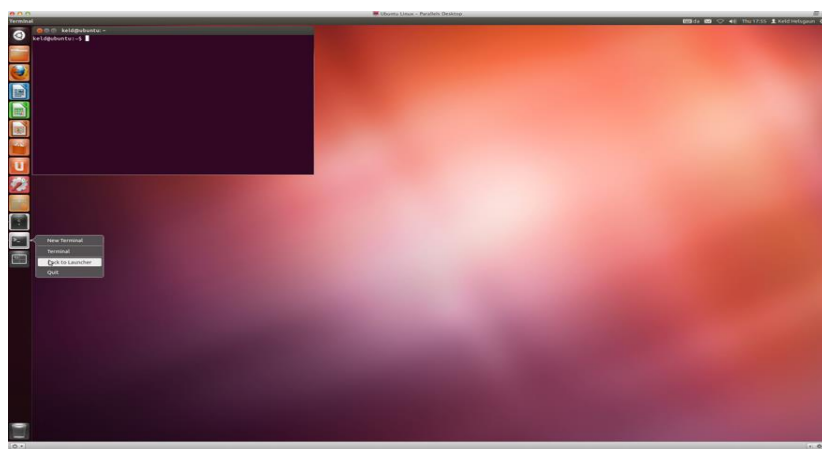
- Dynamic memory management
- Linear data structures and their applications such as stacks, queues and lists
- Non-Linear data structures and their applications such as trees and graphs

Course Outcomes

- Analyze various linear and non-linear data structures
- Demonstrate the working nature of different types of data structures and their applications
- Use appropriate searching and sorting algorithms for the give scenario.
- Apply the appropriate data structure for solving real world problems.

Compile and Execute C Program in Linux operating system :

Step 1: Open up a terminal



Step 2: Use a text editor to create the C source code.

Type the command

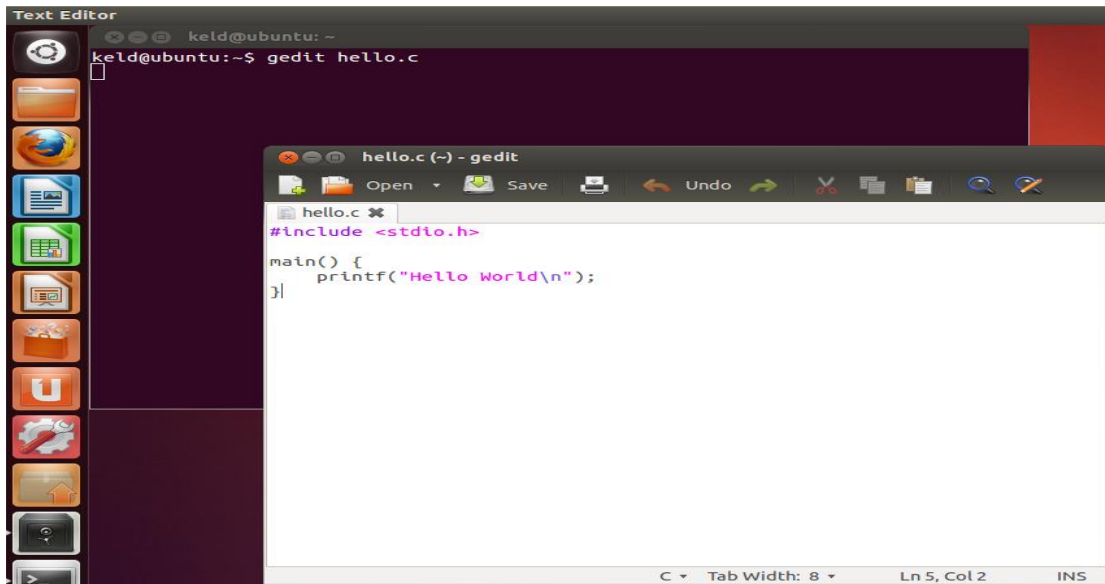
gedit hello.c

and enter the C source code below:

```
#include <stdio.h>  
main()
```

```
{  
    printf("Hello World\n");  
}
```

Close the editor window.

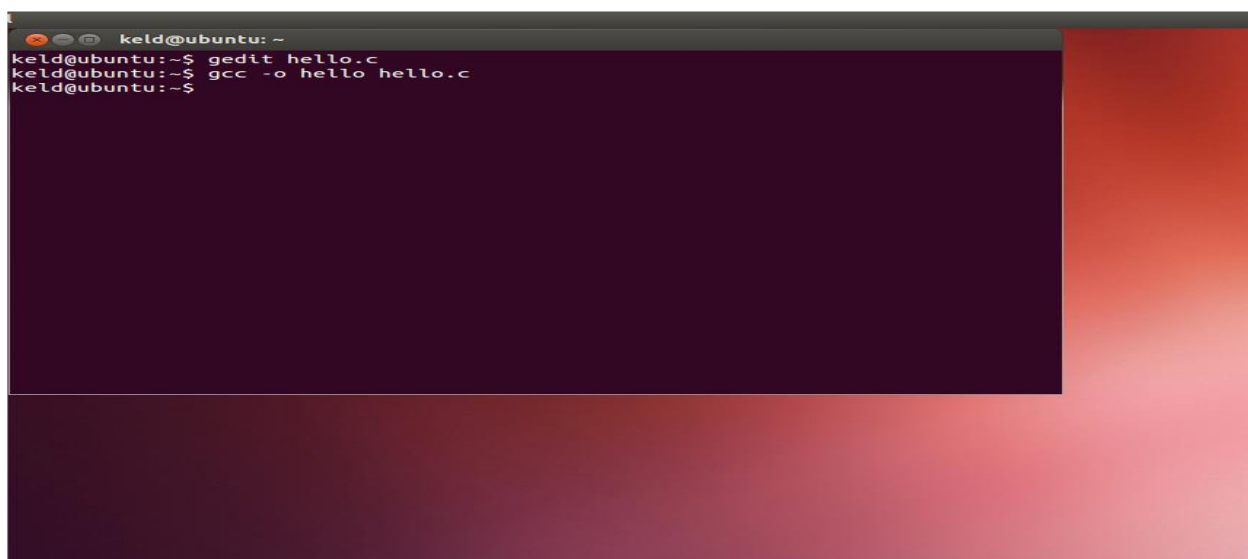


Step 3: Compile the program.

Type the command

```
gcc -o hello hello.c
```

This command will invoke the GNU C compiler to compile the file **hello.c** and output (-o) the result to an executable called **hello**.



Step 4. Execute the program.

Type the command

```
./hello
```

This should result in the output
Hello World

SYLLABUS

Exp.No.	DETAILS	HOURS
1	<p>Develop a Program in C for the following:</p> <p>a) Declare a calendar as an array of 7 elements (A dynamically Created array) to represent 7 days of a week. Each Element of the array is a structure having three fields. The first field is the name of the Day (A dynamically allocated String), The second field is the date of the Day (A integer), the third field is the description of the activity for a particular day (A dynamically allocated String).</p> <p>b) Write functions create(), read() and display(); to create the calendar, to read the data from the keyboard and to print weeks activity details report on screen.</p>	4
2	<p>Develop a Program in C for the following operations on Strings.</p> <p>a. Read a main String (STR), a Pattern String (PAT) and a Replace String (REP)</p> <p>b. Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR. Report suitable messages in case PAT does not exist in STR</p> <p>Support the program with functions for each of the above operations. Don't use Built-in functions.</p>	
3	<p>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)</p> <p>a. Push an Element on to Stack</p> <p>b. Pop an Element from Stack</p> <p>c. Demonstrate Overflow and Underflow situations on Stack</p> <p>d. Display the status of Stack</p> <p>e. Exit</p> <p>Support the program with appropriate functions for each of the above operation</p>	2
4	<p>Design, Develop and Implement a Program in C for the following Stack Applications</p> <p>a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^</p> <p>b. Solving Tower of Hanoi problem with n disks</p>	2
5	<p>Singly Linked List (SLL) of Integer Data</p> <p>a. Create a SLL stack of N integer.</p> <p>b. Display of SLL</p> <p>c. Linear search.</p> <p>Create a SLL queue of N Students Data Concatenation of two SLL of integers.</p>	2

DATA STRUCTURES LABORATORY**BCSL305**

6	Develop a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX) a. Insert an Element on to Circular QUEUE b. Delete an Element from Circular QUEUE c. Demonstrate Overflow and Underflow situations on Circular QUEUE d. Display the status of Circular QUEUE e. Exit Support the program with appropriate functions for each of the above operations	2
7	Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Programme, Sem, PhNo a. Create a SLL of N Students Data by using front insertion. b. Display the status of SLL and count the number of nodes in it c. Perform Insertion / Deletion at End of SLL d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack) e. Exit	2
8	Develop a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: SSN, Name, Dept, Designation, Sal, PhNo 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	2
9	Develop a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes a. Represent and Evaluate a Polynomial $P(x,y,z) = 6x^2y^2z - 4yz^5 + 3x^3yz + 2xy^5z - 2xyz^3$ b. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z) Support the program with appropriate functions for each of the above operations.	2
10	Develop a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers . a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2 b. Traverse the BST in Inorder, Preorder and Post Order c. Search the BST for a given element (KEY) and report the appropriate message d. Exit	2
11	Develop 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 DFS/BFS method.	

DATA STRUCTURES LABORATORY**BCSL305**

12	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. Develop a Program in C that uses Hash function $H:K \rightarrow L$ as $H(K)=K \bmod m$ (remainder method), and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing.	
TOTALHOURS		20

Conduct of Practical Session:

Experiment distribution

o For laboratories having only one part: Students are allowed to pick one experiment from the lot with equal opportunity.

o For laboratories having PART A and PART B: Students are allowed to pick one experiment from PART A and one experiment from PART B, with equal opportunity.

- Change of experiment is allowed only once and marks allotted for procedure to be made zero of the changed part only.

- Marks Distribution (Need to change in accordance with university regulations)

c) For laboratories having only one part – Procedure + Execution + Viva-Voce: $15+70+15 = 100$ Marks

d) For laboratories having PART A and PART B

i. Part A – Procedure + Execution + Viva = $6 + 28 + 6 = 40$ Marks

ii. Part B – Procedure + Execution + Viva = $9 + 42 + 9 = 60$ Marks

PROGRAM 1

Develop a Program in C for the following:

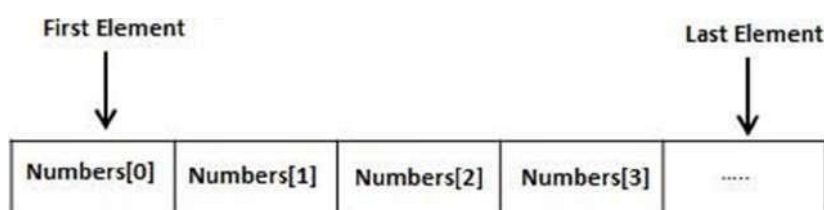
- a) Declare a calendar as an array of 7 elements (A dynamically Created array) to represent 7 days of a week. Each Element of the array is a structure having three fields. The first field is the name of the Day (A dynamically allocated String), The second field is the date of the Day (A integer), the third field is the description of the activity for a particular day (A dynamically allocated String).
- b) Write functions create(), read() and display(); to create the calendar, to read the data from the keyboard and to print weeks activity details report on screen.

Theory:

Arrays are a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1,..., and number 99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



Declaring Arrays

To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows—

```
Type array Name[array Size];
```

This is called a *single-dimensional* array. The **array Size** must be an integer constant greater than zero and **type** can be any valid C data type. For example, to declare a 10-element array called **balance** of type double, use this statement

```
Double balance[10];
```


Here *balance* is a variable array which is sufficient to hold upto 10 double numbers.

Initializing Arrays You can initialize an array in C either one by one or using a single statement as follows—

```
Double balance[5]={ 1000.0,2.0,3.4,7.0,50.0};
```

The number of values between braces { } cannot be larger than the number of elements that we declare for

the array between square brackets [].

If you omit the size of the array, an array just big enough to hold the initialization is created.

Therefore if you write—

```
Double balance[]={ 1000.0,2.0,3.4,7.0,50.0};
```

You will create exactly the same array as you did in the previous example. Following is an example to assign a single element of the array—

```
balance[4]=50.0;
```

The above statement assigns the 5th element in the array with a value of 50.0. All arrays have 0 as the index of their first element which is also called the base index and the last index of an array will be total size of the array minus 1. Shown below is the pictorial representation of the array we discussed above—

	0	1	2	3	4
balance	1000.0	2.0	3.4	7.0	50.0

Accessing Array Elements

An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array. For example—

```
Double salary =balance[9];
```

The above statement will take the 10th element from the array and assign the value to salary variable. The following example Shows how to use all the three above mentioned concepts viz. declaration, assignment, and accessing arrays.

Dynamic Memory Allocation can be defined as a procedure in which the size of a data structure (like Array) is changed during the runtime. C provides some functions to achieve these tasks. There are 4 library functions provided by C

defined under `<stdlib.h>` header file to facilitate dynamic memory allocation in C programming.

C malloc() method

The “**malloc**” or “**memory allocation**” method in C is used to dynamically allocate a single large block of memory with the specified size. It returns a pointer of type void which can be cast into a pointer of any form. It doesn't Initialize memory at execution time so that it has initialized each block with the default garbage value initially.

Syntax of malloc() in C

ptr = (cast-type*) malloc(byte-size)

For Example: `ptr = (int*) malloc(100 * sizeof(int));`

Since the size of int is 4 bytes, this statement will allocate 400 bytes of memory. And, the pointer ptr holds the address of the first byte in the allocated memory.

SOLUTION 1**PROGRAM CODE:**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define WEEK 7
// Structure to represent a day
typedef struct
{
    char *DayName; // Dynamically allocated string for the day name
    int Date;      // Date of the day
    char *Activity; // Dynamically allocated string for the activity description
}DAYTYPE;

void FreeCal(DAYTYPE *);
void DispCal(DAYTYPE *);
void ReadCal(DAYTYPE *);
DAYTYPE *CreateCal();

int main()
{
    // Create the calendar
    DAYTYPE *weeklyCalendar = CreateCal();

    // Read data from the keyboard
    ReadCal(weeklyCalendar);

    // Display the week's activity details
    DispCal(weeklyCalendar);

    // Free allocated memory
    FreeCal(weeklyCalendar);

    return 0;
}

DAYTYPE *CreateCal()
{
    DAYTYPE *calendar = (DAYTYPE *)malloc(sizeof(DAYTYPE) * WEEK);
    for(int i = 0; i < WEEK; i++)
```

```
{
calendar[i].DayName = NULL;
calendar[i].Date = 0;
calendar[i].Activity = NULL;
}
return calendar;
}
void ReadCal(DAYTYPE *calendar)
{
char Choice;
for(int i = 0; i < WEEK; i++)
{
printf("Do you want to enter details for day %d [Y/N]: ", i + 1);
scanf("%c", &Choice); getchar();

if(tolower(Choice) == 'n')
continue;

printf("Day Name: ");
char nameBuffer[50];
scanf("%s", nameBuffer);
calendar[i].DayName = strdup(nameBuffer); // Dynamically allocate and copy the string

printf("Date: ");
scanf("%d", &calendar[i].Date);

printf("Activity: ");
char activityBuffer[100];
scanf(" %[^\n]s", activityBuffer); // Read the entire line, including spaces
calendar[i].Activity = strdup(activityBuffer);

printf("\n");
getchar(); //remove trailing enter character in input buffer
}
}

void DispCal(DAYTYPE *calendar)
{
printf("\nWeek's Activity Details:\n");
for(int i = 0; i < WEEK; i++)
{
printf("Day %d:\n", i + 1);
if(calendar[i].Date == 0)
{
printf("No Activity\n");
continue;
}
```

```
}
```

```
printf(" Day Name: %s\n", calendar[i].DayName);  
printf(" Date: %d\n", calendar[i].Date);  
printf(" Activity: %s\n", calendar[i].Activity);  
}  
}
```

```
void FreeCal(DAYTYPE *calendar)  
{  
for(int i = 0; i < WEEK; i++)  
{  
free(calendar[i].DayName);  
free(calendar[i].Activity);  
}  
free(calendar);  
}
```

output :

Do you want to enter details for day 1 [Y/N]: y

Day Name: monday

Date: 6

Activity: picnic

Do you want to enter details for day 2 [Y/N]: n

Do you want to enter details for day 3 [Y/N]: n

Do you want to enter details for day 4 [Y/N]: n

Do you want to enter details for day 5 [Y/N]: n

Do you want to enter details for day 6 [Y/N]: n

Do you want to enter details for day 7 [Y/N]: y

Day Name: sunday

Date: 3

Activity: sleeping

Week's Activity Details:

Day 1:

Day Name: monday

Date: 6

Activity: picnic

Day 2:

No Activity

Day 3:

No Activity

Day 4:

No Activity

Day 5:

No Activity
Day 6:
No Activity
Day 7:
Day Name: sunday
Date: 3
Activity: sleeping
Exit

SOLUTION 2

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

// Structure to represent a day
typedef struct
{
    char *dayName;    // Dynamically allocated string for the day name
    int date;         // Date of the day
    char *activity;   // Dynamically allocated string for the activity description
}Day;

// Function declarations
Day *createCalendar();
void readData(Day *);
void displayCalendar(Day *);

int main()
{
    // Create the calendar
    Day *weeklyCalendar = createCalendar();

    // Read data from the keyboard
    readData(weeklyCalendar);

    // Display the week's activity details
    displayCalendar(weeklyCalendar);
}
```

```
// Free allocated memory
for (int i = 0; i < 7; i++)
{
    free(weeklyCalendar[i].dayName);
    free(weeklyCalendar[i].activity);
}
free(weeklyCalendar);

return 0;
}

Day *createCalendar()
{
    Day *calendar = (Day *)malloc(7 * sizeof(Day));

    printf("\n% x\n% d",calendar,calendar);

    if (calendar == NULL)
    {
        printf("Memory allocation failed\n");
        exit(0);
    }

    for (int i = 0; i < 7; i++)
    {
        calendar[i].dayName = NULL;
        calendar[i].date = 0;
        calendar[i].activity = NULL;
    }

    return calendar;
}

void readData(Day *calendar)
{
    char nameBuffer[50], activityBuffer[100];

    for (int i = 0; i < 7; i++)
    {
        printf("Enter details for day %d:\n", i + 1);

        printf("Day Name: ");
        scanf("%s", nameBuffer);
```



```
    calendar[i].dayName = strdup(nameBuffer);

    printf("Date: ");
    scanf("%d", &calendar[i].date);

    printf("Activity: ");
    scanf(" %[^\\n]", activityBuffer);
    calendar[i].activity = strdup(activityBuffer);
}
}

void displayCalendar(Day *calendar)
{
    printf("\\nWeek's Activity Details:\\n");
    for (int i = 0; i < 7; i++)
    {
        printf("Day %d:\\n", i + 1);
        printf(" Day Name: %s\\n", calendar[i].dayName);
        printf(" Date: %d\\n", calendar[i].date);
        printf(" Activity: %s\\n\\n", calendar[i].activity);
    }
}
```

PROGRAM -2

2. 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 ReplaceString (REP)

b. Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR. Report suitable messages in case PAT does not exist in STR.

Support the program with functions for each of the above operations. Don't use Built-in functions.*/

SOLUTION 1**ABOUT THE EXPERIMENT:**

Strings are actually one-dimensional array of characters terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null. The following declaration and initialization create a string consisting of the word "Hello".

To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

If you follow the rule of array initialization then you can write the above statement as follows:

```
char greeting[] = "Hello";
```

C language supports a wide range of built-in functions that manipulate null-terminated strings as follows:

`strcpy(s1,s2);` Copies string s2 into string s1.

`strcat(s1,s2);` Concatenates string s2 onto the end of string s1.

`strlen(s1);` Returns the length of string s1.

`strcmp(s1,s2);` Returns 0 if s1 and s2 are the same; less than 0 if s1 < s2.

`strchr(s1,ch);` Returns a pointer to the first occurrence of character ch in string s1.

`strstr(s1,s2);` Returns a pointer to the first occurrence of string s2 in string s1.

ALGORITHM:

Step 1: Start.

Step 2: Read main string STR, pattern string PAT and replace string REP. Step 3: Search / find the pattern string PAT in the main string STR.

Step 4: if PAT is found then replace all occurrences of PAT in main string STR with REP string.

Step 5: if PAT is not found give a suitable error message.

Step 6: Stop.

PROGRAM CODE:

```
#include<stdio.h>
void main ()
{
char STR[100], PAT[100], REP[100], ans[100];
int i, j, c, m, k, flag = 0;
printf ("\nEnter the MAIN string: \n");
gets (STR);
printf ("\nEnter a PATTERN string: \n");
gets (PAT);
printf ("\nEnter a REPLACE string: \n");
gets (REP);
i = m = c = j = 0; while (STR[c] != '\0')
{
// Checking for Match
if (STR[m] == PAT[i])
{
i++; m++;
flag = 1;
if (PAT[i] == '\0')
{
//copy replace string in ans string
for (k = 0; REP[k] != '\0'; k++, j++)
ans[j] = REP[k]; i = 0;
c = m;
}
}

else //mismatch
{
ans[j] = STR[c];
j++; c++;
m = c;
i = 0;
}
}
if (flag == 0)
{
printf ("Pattern doesn't found!!!");
}
else
{
```

```
ans[j] = '\0';  
printf ("\nThe RESULTANT string is:%s\n", ans);  
}  
}
```

Sample Output 1

Enter the MAIN string: good morning
Enter a PATTERN string: morning
Enter a REPLACE string: evening
The RESULTANT string is: good evening

Sample Output 2

Enter the MAIN string: hi vcet

Enter a PATTERN string: bye

Enter a REPLACE string: hello Pattern doesn't found!!

SOLUTION 2**PROGRAMCODE:**

```
#include<stdio.h>  
#include<conio.h>  
  
//Declarations  
char str[100], pat[50], rep[50], ans[100]; inti, j, c, m, k, flag=0;  
void string match()  
{  
i = m = c = j = 0; while(str[c]!='\0')  
{  
if(str[m]==pat[i]) // matching  
{ i++; m++;  
if(pat[i]!='\0') // found occurrences.  
{  
flag=1;  
  
// copy replace string in an string.  
for(k=0;rep[k]!='\0';k++,j++)  
ans[j]=rep[k];  
i = 0;
```

```

c=m;
    }
}    //ifends.
else    //...mismatch
{
ans[j] = str[c]; j++;
c++;
m=c; i = 0;
    }//elseends
} //endofwhile ans[j]='\0';
} //end string match()

```

```

Void main()
{
clrscr();
printf("\nEnter a main string \n");
gets(str);
printf("\nEnter a pattern string \n"); flushall();
gets(pat);
printf("\nEnter a replace string \n"); flushall();
gets(rep); stringmatch();if(flag==1) else
printf("\nThe resultant string is\n%s", ans); printf("\nPattern string NOT found\n"); getch();
}    //end ofmain

```

SAMPLEOUTPUT:

RUN1:

Enter a main string:Test Enter a pattern string:Te Enter a replace string :Re

The resultant string is Rest

RUN2:

Enter a main string:This is Data Structure lab Enter a pattern string:DataStructure
Enter a replace string:Datastructure with C

The resultant string is This is Data structure with C lab

RUN3:

Enter a main string:This is Data Structure lab Enter a pattern string:Date
Enter a replace string:DATA Pattern string NOT found

SOLUTION 3

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

int main()
{
    char str[100],pat[100],rep[100], ans[100]; int i,j,k,c,m,flag=0;
    printf("Enter the main string: "); gets(str);
    printf("Enter the pattern string: "); gets(pat);
    printf("Enter the replace string: "); gets(rep);
    i=j=m=c=0; while(str[c]!='\0')
    {
        if(str[m]==pat[i])
        {
            i++; m++;
            if(pat[i]!='\0')
            {
                for(k=0;rep[k]!='\0';k++,j++) ans[j] = rep[k];
                i=0;
                c=m; flag=1;
            }
        }
        else{
            ans[j] = str[c]; j++;
            c++;
            m=c; i=0;
        }
    }
    if(flag==0)
    {
        printf("\n\n\tpattern %s not found",pat);
    }
    else{
        ans[j] = '\0';
        printf("\n\t Resultant string is %s",ans[j]);
    }
    return 0;
}
```

SAMPLE OUTPUT

Case i)

Enter the MAIN string:

Good Morning

Enter a PATTERN string:

Morning

Enter a REPLACE string:

Evening

The RESULTANT string is: Good Evening

Case ii)

Enter the MAIN string:

Good Morning

Enter a PATTERN string:

Noon

Enter a REPLACE string:

Noon

Pattern doesn't found!!!

PROGRAM-3

3. 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)

Push an element on to stack Pop an element from stack. Demonstrate how stack can be used to check palindrome. Demonstrate Overflow and Underflow situations on stack. Display the status of stack.

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

ABOUT THE EXPERIMENT:

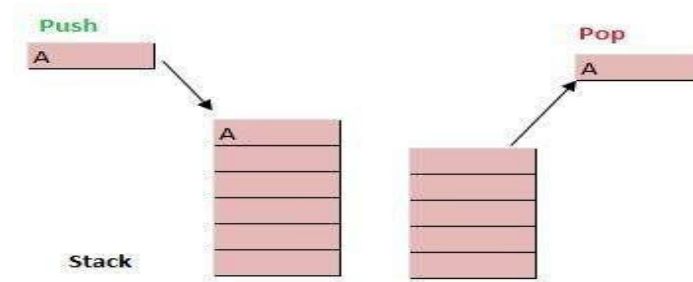
A stack is an abstract data type (ADT), commonly used in most programming languages. It is named stack as it behaves like a real-world stack.



A real-world stack allows operations at one end only. For example, we can place or remove a card or plate from top of the stack only. Likewise, Stack ADT allows all data operations at one end only. At any given time, we can only access the top element of a stack.

This feature makes it LIFO data structure. LIFO stands for Last-in-first-out. Here, the element which is placed (inserted or added) last is accessed first. In stack terminology, insertion operation is called **PUSH** operation and removal operation is called **POP** operation.

Below given diagram tries to depict a stack and its operation



A stack can be implemented by means of Array, Structure, Pointer and Linked-List. Stack can either be a fixed size one or it may have a sense of dynamic resizing. Here, we are going to implement stack using arrays which makes it a fixed size stack implementation.

Basic Operations:

push()-pushing(storing)an element on the stack.

pop() -removing(accessing) an element from the stack.

To use a stack efficiently we need to check status of stack as well. For the same purpose, the following functionality is added to stacks;

peek()—get the top data element of the stack, without removing it.

isFull() —check if stack is full.

isEmpty()—check if stack is empty.

SOLUTION 1

ALGORITHM:

Step1:Start.

Step2:Initialize stack size MAX and top of stack-1.

Step3:Push integer element onto stack and display the contents of the stack .if stack is full give a messageas 'Stack is Overflow'.

Step3:Pop element from stack along with display the stack contents. if stack is empty give a messageas 'Stack is Underflow'.

Step4::Stop.

PROGRAMCODE:

```
#include<stdio.h>
#include<conio.h>
#defineMAX4
Intstack[MAX],item;
Intch, top =-1, count=0, status =0;
/*PUSHFUNCTION*/
void push(int stack[],intitem)
{
    if(top ==(MAX-1))
        printf("\n\nStack Is Overflow");
    else
    {
        stack[++top] = item;
        status++;
    }
}

/*POPFUNCTION*/
intpop(int stack[])
{
    intret;
    if(top==-1)
        printf("\n\nStack is Underflow");
    else
    {
        ret = stack[top--];
        status--;
        printf("\nPopped element is %d",ret);
    }

    returnret;
}

/*FUNCTION TO DISPLAY STACK*/
Void display(int stack[])
{
    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");
        }
    }
```

```
/*MAINPROGRAM*/
```

```
voidmain()
{
    clrscr();
    do{
        printf("\n\n --- MAIN MENU \n");
        printf("\n1. PUSH (Insert) in the Stack");
        printf("\n2. POP (Delete) from the Stack");
        printf("\n3. 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(stack,item);
                    display(stack);
                    break;
            case2:item=pop(stack);
                    display(stack);
                    break;
            case3:exit(0);
                    break;
            default:printf("\nEND OF EXECUTION");
        }
    }//end switch
    while(ch!=4);
    getch();
}
```

SAMPLEOUTPUT:

----MAINMENU----

- 1.PUSH (Insert) In the Stack
- 2.POP (Delete) from the Stack
- 3.Exit (End the Execution)

Enter Your Choice: 1

Enter an element to be pushed: 1

The stack contents are:

|1|

----MAINMENU----

- 1.PUSH(Insert) in th eStack
- 2.POP (Delete) from theStack
- 3.Exit (End the Execution)

Enter You rChoice: 1

Enter an element to be pushed: 2

The stack contents are:

|2|

|1|

-----AFTER THE 4 TIMES PUSH OPERATION

----MAIN MENU----

- 1.PUSH(Insert) in the Stack
- 2.POP (Delete) from the Stack
- 3.Exit (End the Execution)

EnterYourChoice: 1

Enter an element to be pushed: 9

StackisOverflow

The stack contents are:

|1|

|2|

|2|

|1|

----MAIN MENU----

- 1.PUSH(Insert) in the Stack
- 2.POP (Delete) from the Stack
- 3.Exit(End the Execution)

EnterYour Choice: 2

Popped element is: 1

The stack contents are

|2|

|2|

|1|

(-----AFTER THE 4 TIMES POP OPERATION)

----MAIN MENU----

- 1.PUSH(Insert) in the Stack
- 2.POP (Delete) from the Stack
- 3.Exit (End the Execution)

EnterYour Choice: 2

Stack is Underflow

The stack contents are:Stack is Empty

----MAINMENU----

- 1.PUSH(Insert) in the Stack
- 2.POP (Delete)from the Stack
- 3.Exit (End the Execution)

EnterYour Choice: 1

Enter an element to be pushed: 2

The stack contents are:

|2|

|1|

----MAIN MENU----

- 1.PUSH(Insert) in the Stack
- 2.POP (Delete) from the Stack
- 3.Exit (End the Execution)

EnterYour Choice: 1

Enter a element to be pushed: 1

The stack contents are:

|1|

|2|

|1|

SOLUTION 2**PROGRAM CODE:**

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#define max_size 5
int stack[max_size], top = -1, flag = 1;
int i, temp, item, rev[max_size], num[max_size];
void push ();
void pop ();
void display ();
void pali ();
int main ()
{
    int choice;
    printf ("\n\n-----STACK OPERATIONS      ");
    printf ("1.Push\n");
    printf ("2.Pop\n");
    printf ("3.Palindrome\n");
    printf ("4.Display\n");
    printf ("5.Exit\n");
    printf (" ");
    while (1)
    {
        printf ("\nEnter your choice:\t");
        scanf ("%d", &choice);
        switch (choice)
        {
            case 1:
                push ();
                break;
            case 2:
                pop ();
                if (flag)
                    printf ("\nThe popped element: %d\t", item);
                temp = top;
                break;
            case 3:
                pali ();
                top = temp;
                break;
            case 4:
                display ();
                break;
            case 5:
                exit (0);
                break;
            default:
                printf ("\nInvalid choice:\n");
                break;
        }
    }
}
```



```
    }

}

//return 0;
}

void push ()                                //Inserting element into the stack
{
    if (top == (max_size - 1))
    {
        printf ("\nStack Overflow:");
    }
    else
    {
        printf ("Enter the element to be inserted:\t");
        scanf ("%d", &item);
        top = top + 1;
        stack[top] = item;
    }
    temp = top;
}

void pop ()                                //deleting an element from the stack
{
    if (top == -1)
    {
        printf ("Stack Underflow:");
        flag = 0;
    }
    else
    {
        item = stack[top];
        top = top - 1;
    }
}

void pali ()
{
    i = 0;
    if (top == -1)
    {
        printf ("Push some elements into the stack first\n");
    }
    else
    {
        while (top != -1)
        {
            rev[top] = stack[top];
            pop ();
        }
        top = temp;
        for (i = 0; i <= temp; i++)
        {
            if (stack[top--] == rev[i])
```

```

        {
            if (i == temp)
            {
                printf ("Palindrome\n");
                return;
            }
        }
    }
    printf ("Not Palindrome\n");

}

void display ()
{
    int i;
    top = temp;
    if (top == -1)
    {
        printf ("\nStack is Empty:");
    }
    else
    {
        printf ("\nThe stack elements are:\n");
        for (i = top; i >= 0; i--)
        {
            printf ("%d\n", stack[i]);
        }
    }
}

```

Sample Output 1

-----STACK OPERATIONS-----

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

Enter your choice: 1
Enter the element to be inserted: 1

Enter your choice: 1
Enter the element to be inserted: 2

Enter your choice: 1
Enter the element to be inserted: 1

Enter your choice: 1
Enter the element to be inserted: 5

Enter your choice: 2
The popped element: 5

Enter your choice: 4
The stack elements are: 1 2
1
Enter your choice: 3
Numbers= 1
Numbers= 2
Numbers= 1
reverse operation :
reverse array : 1 2 1
check for palindrome : It is palindrome number

Enter your choice: 5
Exit

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

Enter your choice: 1
Enter the element to be inserted: 10

Enter your choice: 1
Enter the element to be inserted: 20

Enter your choice: 1
Enter the element to be inserted: 10

Enter your choice: 1
Enter the element to be inserted: 50

Enter your choice: 2 The popped element: 50

Enter your choice: 4
The stack elements are: 10 20
10
Enter your choice: 3
Numbers= 1
Numbers= 2
Numbers= 1
reverse operation : reverse array : 1 2 1
check for palindrome : It is palindrome number

Enter your choice: 5
Exit

SOLUTION 3

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h> #define
max_size 5
int stack[max_size],top=-1;
void push();
void pop();
void display();
void pali();
int main()
{
    while(choice)
    {
        printf("\n\n-----STACK OPERATIONS ----- \n");
        printf("1.Push\n");
        printf("2.Pop\n");
        printf("3.Palindrome\n");
        printf("4.Display\n");
        printf("5.Exit\n");
        printf("----- ");
        printf("\nEnter your choice:\t");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                push();
                break;
            case 2:
                pop();
                break;
            case 3:
                pali();
                break;
            case 4:
                display();
                break;
            case 5:
                exit(0);
                break;
            default:
```

```

        printf("\nInvalid choice:\n");
        break;
    }
}

return 0;
}

void push() //Inserting element into the stack
{
    int item;
    if(top==(max_size-1))
    {
        printf("\nStack Overflow:");
    }
    else
    {
        printf("Enter the element to be
        inserted:\t"); scanf("%d",&item);
        top=top+1;
        stack[top]=item;
    }
}

void pop() //deleting an element from the stack
{
    int item;
    if(top== -1)
    {
        printf("Stack Underflow:");
    }
    else
    {
        item=stack[top];
        top=top-1;
        printf("\nThe popped element: %d\t",item);
    }
}

void pali()
{
    int digit,j,k,len=top+1,ind=0;
    int num[len],rev[len],i=0;
    while(top!=-1)
    {

```

```

        digit= stack[top];
        num[i]=digit;
        top--;
        i++;
    }
    for(j=0;j<len;j++)
    {
        printf("Numbers= %d\n",num[j]);
    }
    printf("reverse operation : \n");
    for(k=len-1;k>=0;k--)
    {
        rev[k]=num[ind];
        ind++;
    }
    printf("reverse array : ");
    for(k=0;k<len;k++)
    {
        printf("%d\n",rev[k]);
    }
    printf("check for palindrome: \n");
    int length = 0;
    for(i=0;i<len;i++) {
        if(num[i]==rev[i])
        { length = length+1;
        }
    }
    if(length==len)
    {
        printf("It is palindrome number\n");
    }
    else
    {
        printf("It is not a palindrome number\n");
    }
    top = len-1;
}

void display()
{
    int i;
    if(top== -1)

```

```

{
printf("\nStack is Empty:");
}
else
{
printf("\nThe stack elements are:\n" );
for(i=top;i>=0;i--)

{
printf("%d\n",stack[i]);
}
}
}

```

SAMPLE Output:

-----STACK OPERATIONS-----

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

Enter your choice:1

Enter the element to be inserted: 1

-----STACK OPERATIONS-----

- 1.Push
- 2.Pop
- 3.Palindrome

4.Display

5.Exit

Enter your choice: 1

Enter the element to be inserted: 2

-----STACK OPERATIONS-----

- 1.Push
- 2.Pop
- 3.Palindrome

4.Display

5.Exit

Enter your choice: 1

Enter the element to be inserted: 1

-----STACK OPERATIONS-----

1.Push

2.Pop

3.Palindrome

4.Display

5.Exit

Enter your choice: 4

The stack elements are: 1 2 1

-----STACK OPERATIONS-----

1.Push

2.Pop

3.Palindrome

4.Display

5.Exit

Enter your choice: 4

The stack elements are:

1

2

1

-----STACK OPERATIONS-----

1.Push

2.Pop

3.Palindrome

4.Display

5.Exit

Enter your choice: 3

Numbers= 1

Numbers= 2
Numbers= 1
reverse operation :
reverse array :
1
2
1
check for palindrome :
It is palindrome number

-----STACK OPERATIONS-----

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

Enter your choice: 2

The popped element: 1

-----STACK OPERATIONS-----

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

Enter your choice: 4

The stack elements are: 2 1

-----STACK OPERATIONS-----

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

Enter your choice: 1

Enter the element to be inserted: 3

-----STACK OPERATIONS-----

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

Enter your choice: 4

The stack elements are: 3 2 1

-----STACK OPERATIONS-----

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

Enter your choice: 3

Numbers= 3

Numbers= 2

Numbers= 1

reverse operation : reverse

array :

1

2

3

check for palindrome :

It is not a palindrome number

-----STACK OPERATIONS-----

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

Enter your choice: 5

PROGRAM- 4

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 alpha numeric operands.

ABOUTTHEEXPERIMENT:

Infix: Operators are written in-between their operands. Ex: X + Y

Prefix: Operators are written before their operands. Ex: +X Y

postfix: Operators are written after their operands.Ex:XY+

Examples of Infix, Prefix, and Postfix

InfixExpression	PrefixExpression	PostfixExpression
A+B	+AB	AB +
A+B*C	+A*B C	ABC*+

Infix to prefix conversion Expression = (A+B^C)*D+E^5

Step1. Reverse the infix expression= $5^E+D*(C^B+A)$

Step2. Make Every '(' as ')' and every ')' as '(' ($5^E+D*(C^B+A)$)

Step3. Convert expression to postfix form.

Step4. Reverse the expression.

+*+A^BCD^E

Step5. Result

+*+A^BCD^E5

A+(B*C-(D/E-F)*G)*H			
Expression	Stack	Output	Comment
$5^E+D*(C^B+A)$	Empty	-	Initial
$^E+D*(C^B+A)$	Empty	5	Print
$E+D*(C^B+A)$	^	5	Push
$+D*(C^B+A)$	^	5E	Push
$D*(C^B+A)$	+	5E^	PopAndPush
$*(C^B+A)$	+	5E^D	Print
(C^B+A)	+	5E^D	Push
$C^B+A)$	+(5E^D	Push
$^B+A)$	+(5E^DC	Print
$B+A)$	+(5E^DC	Push
$+A)$	+(5E^DCB	Print
$A)$	+(5E^DCB^	PopAndPush
)	+(5E^DCB^A	Print
End	+	5E^DCB^A+	Pop Until '('

SOLUTION 1**ALGORITHM:**

Step 1:Start.

Step2:Read an infix expression with parenthesis and without parenthesis.

Step3:convert the infix expression to postfix expression.

Step 4:Stop

PROGRAMCODE:

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
IntF(charsymbol)
{
    switch(symbol)
    {
        case '+':
        case '-':
            return 2;
        case '*':
        case '/':
            return 4;
        case '^':
        case '$':
            return 5;
        case '(':
            return 0;
        case '#':
            return -1;
        default: return 8;
    }
}
```

```
Int G(char symbol)
{
    switch(symbol)
    {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
```

```

        return 3;
        case '^':
        case '$':return 6;
        case '(':
        return 9;
        case ')':
        return 0;
        default:return 7;
    }
}

Void infix_postfix(char infix[],char postfix[])
{
    Int top, j, i;
    chars[30],symbol;
    top=-1;
    s[++top]='#';
    j = 0;
    for(i=0;i <strlen(infix);i++)
    {
        symbol=infix[i];while(F(s[top])>G(symbol))
        {
            postfix[j] = s[top--];
            j++;
        }
        if(F(s[top]) != G(symbol))
            s[++top]=symbol;
        else
            top--;
    }
    while(s[top]!='#')
    {
        postfix[j++]=s[top--];
    }
    postfix[j]='\0';
}

void main()
{
    char infix[20], postfix[20];
    printf("\nEnter a valid infix expression\n");
    fflush();
    gets(infix);
    infix_postfix(infix,postfix);
    printf("\nThe infix expression is:\n");
    printf("%s",infix);
    printf("\nThe postfix expression is:\n");
    printf("%s",postfix);
    getch();
}

```

}

SAMPLEOUTPUT:

Enter a valid infix expression

(a+(b-c)*d)

The infix expression is:

(a+(b-c)*d)

The postfix expression is:

abc-d*+

SOLUTION 2

```
#define SIZE 50 /* Size of Stack */
#include <ctype.h>
#include <stdio.h>
char s[SIZE];
int top = -1; /* Global declarations */
push(char elem) /* Function for PUSH operation */
{
    s[++top] = elem;
}
char pop() /* Function for POP operation */
{
    return (s[top--]);
}
int pr(char elem) /* Function for precedence */
{
    switch (elem)
    {
        case '#': return 0;
        case '(': return 1;
        case '+':
        case '-': return 2;
        case '*':
        case '/':
        case '%': return 3;
        case '^': return 4;
    }
}
void main() /* Main Program */
{
    char infix[50], postfix[50], ch, elem;
    int i = 0, k = 0;
    printf("\n\nRead the Infix Expression ? ");
    scanf("%s", infix);
    push('#');
    while ((ch = infix[i++]) != '\0')
    {
        if (ch == '(')
            push(ch);
        else if (isdigit(ch))
            postfix[k++] = ch;
        else if (ch == ')')
        {
            while (s[top] != '(')
                postfix[k++] = pop();
            elem = pop(); /* Remove ( */
        }
    }
}
```



```

else /* Operator */
{
    while (pr(s[top]) >= pr(ch))
        pofx[k++] = pop();
    push(ch);
}
}
while (s[top] != '#') /* Pop from stack till empty */
    pofx[k++] = pop();
pofx[k] = '\0'; /* Make pofx as valid string */
printf("\n\nGiven Infix Expn: %s Postfix Expn: %s\n", infix, pofx);
}

```

Sample Output 1

Read the Infix Expression

(a+b)*c/d^5%1

Given Infix Expn:

(a+b)*c/d^5%1

Postfix Expn:

ab+c*d5^/1%

Sample Output 2

Read the Infix Expression

(a+(b-c)*d)

Given Infix Expn:

(a+(b-c)*d)

Postfix Expn:

abc-d*+

SOLUTION 3

Algorithm F(symbol)

```
switch
case+/-return2
case%/*//return4
case^/$return5
case)return0
case#return-1
default:return8
```

Algorithm G(symbol)

```
switch
case+/-return1
case%/*//return3
case^/$return6
case)return0
case)return9
defaultreturn7
```

Algorithm main

```
1.Start
2.Readinfixexpression
3.Checkforvalidexpressionif(infix[i]=='(')
    c++
    if(infix[i]==')')
        c--
4.if(c=0)
    Thencallinfix_postfix(infix,postfix)
5.printthepostfixexpression
else
    printinvalidexpression
6.stop
```

Algorithm infix_postfix(in,po)

```
1.Pushtostack
2.Repeati=0tostringlengthof infix expression
symbol=in[i]
repeatF(s[top])>G(symbol)
po[j++]=s[top--]
if(F(s[top])>G(symbol)
thenpushsymboltostack
else
pop
3.repeatuntilitis'#'
po[j++]=s[top--]
4.po[j]='\0'
```

PROGRAMCODE

```
#include<stdio.h>
Void infix_to_postfix();
Void push(char);
Char pop();
Int priority(char);
Char infix[30],postfix[30],stack[30];
int top=-1;

void main()
{
    printf("Enter the valid Infix expression\n");
    scanf("%s",infix);
    infix_to_postfix();
    printf("\nInfix expression:%s",infix);
    printf("\nPostfix expression:%s\n",postfix);
}

//pushsymboltostack
Void push(char item)
{
    stack[++top]=item;
} //end of function push

//popsymbolfromstack
char pop()
{
    Return stack[top--];
} //end of function pop

//check the priority of operator
Int priority(char symb)
{
    Int p;
    switch(symb)
    {
        case '+':
        case '-':p=1;
                break;
        case '*':
        case '/':
        case '%':p=2;
                break;
        case '^':
        case '$':p=3;
                break;
```

```

        case '(':
        case ')': p=0;
                break;
        case '#': p=-1;
        break;
    } //end of switch return p;
} //end of function priority

//converting an Infix Expression to Postfix Expression
void infix_to_postfix()
{
    inti=0,j=0;
    char symb,temp;
    push('#');
    for(i=0;infix[i]!='\0';i++)
    {
        symb=infix[i];switch(symb)
        {
            case '(': push(symb); //push to top of stack
                break;
            case ')': temp=pop(); //pop symbol from top of stack
                while(temp!='(') //pop all symbols from top of stack
                {
                    postfix[j++]=temp;
                    temp=pop();
                } //end of while
                break;
            case '+':
            case '-':
            case '*':
            case '/':
            case '%':
            case '^':
            case '$':
            while(priority(stack[top])>=priority(symb)) //check for priority of operator //
            {
                temp=pop();
                postfix[j++]=temp;
            }
            push(symb);
            break;
            default:
                postfix[j++]=symb;
        } //end of switch
    }
}

```

```
        } //end of for
while(top>0)
{
    temp=pop();
    postfix[j++]=temp;
} //end of while
    postfix[j]='\0'; //end string postfix
} //end of function infix_to_postfix
```

SAMPLE OUTPUT:

Enter the vaild infix expression

a-b+c

infix expression: a-b+c

postfix expression: ab-c+

Enter the vaild infix expression

a+(b+c*d)+c

infix expression: a+(b+c*d)+c

postfix expression: abcd*++c+

PROGRAM -5

5.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.

ABOUT THE EXPERIMENT:

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

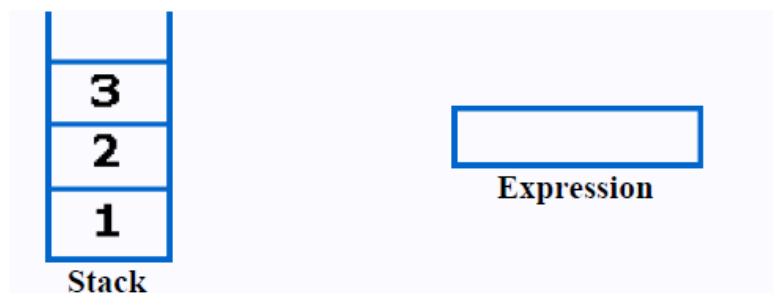
Postfix/Suffix Expression: Operators are written after their operands. Ex:XY+

In normal algebra we use the infix notation like $a+b*c$.

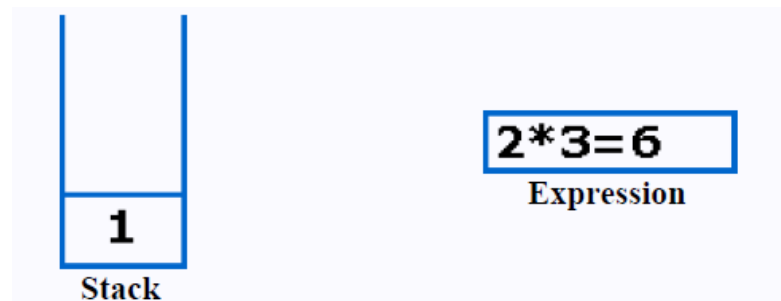
The corresponding postfix notation is $abc*+$

Example: Postfix String: 123*+4-

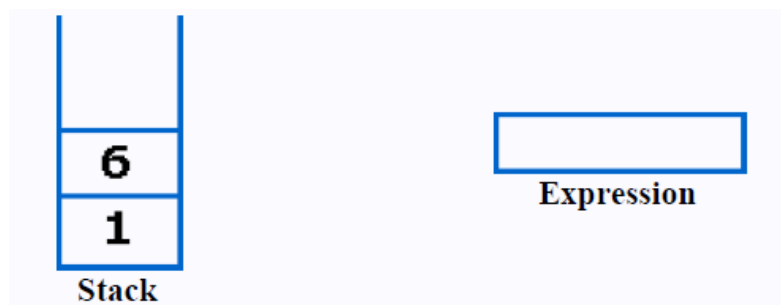
Initially the Stack is empty. Now, the first three characters scanned are 1,2 and 3, which are operands. Thus they will be pushed into the stack in that order.



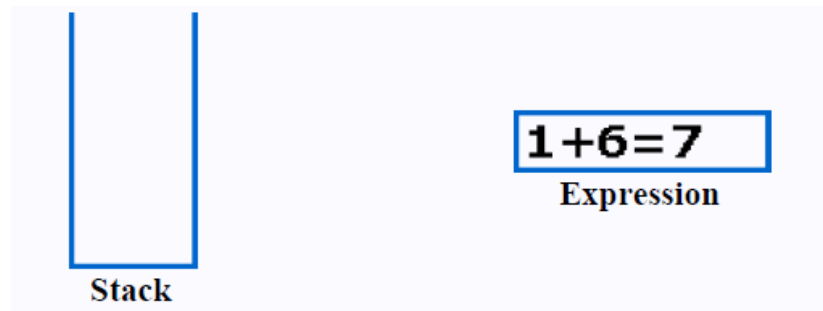
Next character scanned is "*", which is an operator. Thus, we pop the top two elements from the stack and perform the "*" operation with the two operands. The second operand will be the first element that is popped.



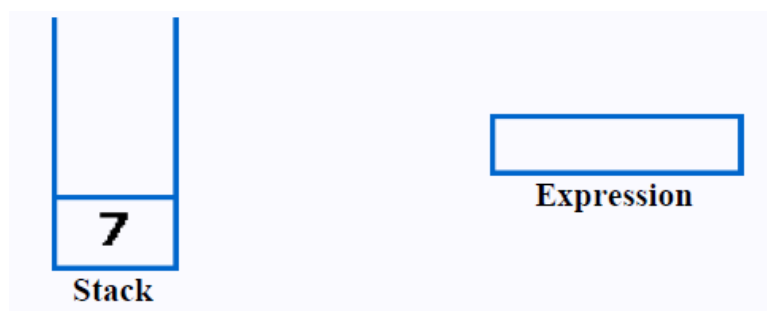
The value of the expression (2*3) that has been evaluated (6) is pushed into the stack.



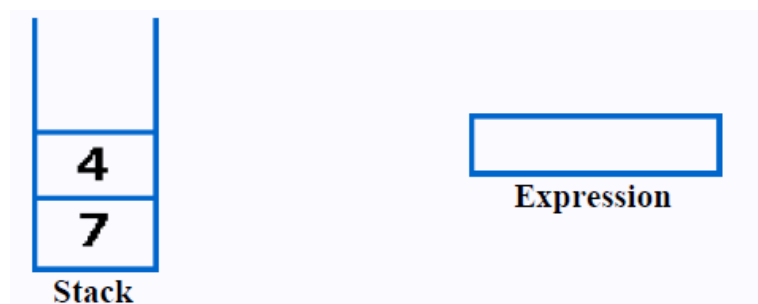
Next character scanned is "+", which is an operator. Thus, we pop the top two elements from the stack and perform the "+" operation with the two operands. The second operand will be then first element that is popped.



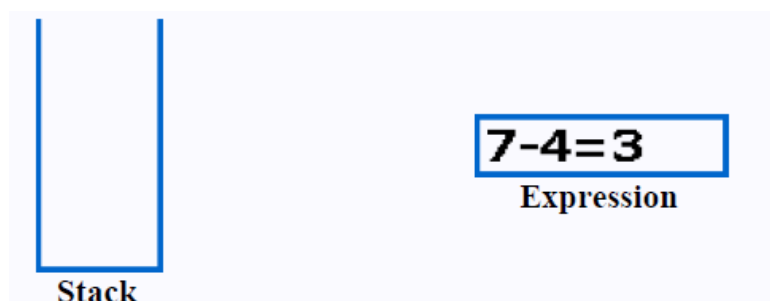
The value of the expression (1+6) that has been evaluated (7) is pushed into the stack.



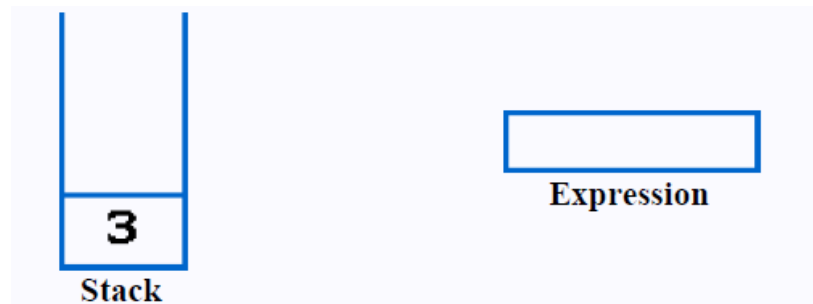
Next character scanned is "4", which is added to the stack.



Next character scanned is "-", which is an operator. Thus, we pop the top two elements from the stack and perform the "-" operation with the two operands. The second operand will be the first element that is popped.



The value of the expression(7-4) that has been evaluated(3) is pushed into the stack.



Now,since all the characters are scanned, the remaining element in the stack (there will be only one element in the stack)will be returned.

End	
Postfix String: 123*+4	

ALGORITHM:

Step 1:Start.

Step 2:Read the postfix/suffix expression.

Step 3:Evaluate the postfix expression based on the precedence of the operator.

Step 4:Stop.

SOLUTION 1

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int i,top=-1;
int op1, op2, res,s[20];
char postfix[90],symb;
```

```
void push(int item)
{
    top = top+1;
    s[top]= item;
}
```

```
int pop()
{
    int item;
    item=s[top];
    top = top-1;
    return item;
}
```

```
void main()
{
```



```

printf("\nEnter a valid postfix expression:\n");
scanf("%s",postfix);
for(i=0;postfix[i]!='\0';i++)
{
    symb= postfix[i];
    if(isdigit(symb))
    {
        push(symb-'0');
    }
    else
    {
        op2 = pop();
        op1 = pop();
        switch(symb)
        {
case'+':
            push(op1+op2);
            break;
case'-':
            push(op1-op2);
            break;
case'*':
            push(op1*op2);
            break;
case'/':
            push(op1/op2);
            break;
case'%':
            push(op1%op2);
            break;
case'$':
            case'^':push(pow(op1,op2));
            break;
default:push(0);
        }
    }
}
res= pop();
printf("\nResult= %d",res);
}

```

Output:

Enter a valid postfix expression:
623+-382/+*2\$3+
Result=52

Enter a valid postfix expression:
42\$3*3-84/11+/>

Result=46

SOLUTION 2

```
#include<stdio.h>
#define MAX 20
typedef struct stack
{
    int data[MAX];
    int top;
}stack;
void init(stack *);
int empty(stack *);
int full(stack *);
int pop(stack *);
void push(stack *,int);
int evaluate(char x,int op1,int op2);
int main()
{
    stack s;
    char x;
    int op1,op2,val;
    init(&s);
    printf("Enter the expression: Single digit operand and operators only:");
    while((x=getchar())!='\n')
    {
        if(isdigit(x))
            push(&s,x-48); //x-48 for removing the effect of ASCII
        else
        {
            op2=pop(&s);
            op1=pop(&s);
            val=evaluate(x,op1,op2);
            push(&s,val);
        }
    }
    val=pop(&s);
    printf("\nValue of expression=%d",val);

    return 0;
}

int evaluate(char x,int op1,int op2)
{
    if(x=='+')
        return(op1+op2);
    if(x=='-')
        return(op1-op2);
}
```

```

if(x=='*')
return(op1*op2);
if(x=='/')
return(op1/op2);
if(x=='%')
return(op1%op2);
}

void init(stack *s)
{
s->top=-1;
}

int empty(stack *s)
{
    if(s->top==-1)
        return(1);
    return(0);
}

int full(stack *s)
{
    if(s->top==MAX-1)
        return(1);
    else
        return(0);
}

void push(stack *s,int x)
{
    s->top=s->top+1;
    s->data[s->top]=x;
}

int pop(stack *s)
{
    int x;
    x=s->data[s->top];
    s->top=s->top-1;
    return(x);
}

```

Output

Enter the expression(eg: 59+3*)
 Single digit operand and operators only:74+5-
 Value of expression=6

SOLUTION 3

```
#include <stdio.h>
#include <stdlib.h>
#include<math.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);
    }
}
void main()
{
    double s[20];
    double res;
    double op1;
    double op2;
    int top,i;
    char postfix[20];
    char symbol;

    printf("enter 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("3the result is: %f",res);
}

```

Output:

Insert a postfix notation :: 45^98*+

Result :: 1096

Process returned 16 (0x10) execution time : 8.469 s

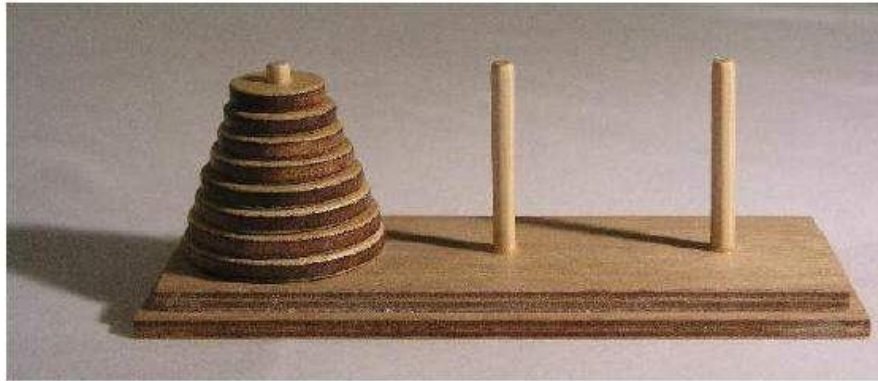
Press any key to continue.

b. Solving Tower of Hanoi problem with n disks.

The towers of Hanoi is a mathematical game or puzzle. It consists of three rods and number of disks of different size which can slide onto any rod. The puzzle starts with the disks in stacked in ascending order of size one rod, the smallest at the top thus making a conical shape. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules

- Only one disk can be moved at a time
- Each move consists of taking the upper disk from one of the stack and placing it on top of another stack.
- No disk may be placed on top of a smaller disk.

With three disks, the puzzle can be solved in seven moves. The minimum number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$, where n is the number of disks.



ALGORITHM:

Step 1: Start.

Step 2: Read N number of discs.

Step 3: Move all the discs from source to destination by using temp rod.

Step 4: Stop.

SOLUTION 1

```
#include <stdio.h>
void towers(int, char, char, char);
int main()
{
    int num;
    printf("Enter the number of disks : ");
    scanf("%d", &num);
    printf("The sequence of moves involved in the Tower of Hanoi are :\n");
    towers(num, 'A', 'C', 'B');
    return 0;
}
void towers(int num, char frompeg, char topeg, char auxpeg)
{
    if (num == 1)
    {
        printf("\n Move disk 1 from peg %c to peg %c", frompeg, topeg);
        return;
    }
    towers(num - 1, frompeg, auxpeg, topeg);
    printf("\n Move disk %d from peg %c to peg %c", num, frompeg, topeg);
    towers(num - 1, auxpeg, topeg, frompeg);
}
```

```
}
```

Output:

Enter the number of disks: 4

The sequence of moves involved in the Tower of Hanoi are:

Move disk 1 from peg A to peg B
Move disk 2 from peg A to peg C
Move disk 1 from peg B to peg C
Move disk 3 from peg A to peg B
Move disk 1 from peg C to peg A
Move disk 2 from peg C to peg B
Move disk 1 from peg A to peg B
Move disk 4 from peg A to peg C

SOLUTION 2

```
#include<stdio.h>
void towers_hanoi(int n)
{
    int x;
    for(x=1;x<(1<<n);x++)
        printf("Move from Pole %d to Pole %d\n", (x&(x-1))%3, (((x|(x-1)) + 1)%3));
}

int main()
{
    int m;
    printf("\n Enter the Number of Disks : ");
    scanf("%d",&m);
    towers_hanoi(m);
    return 0;
}
```

SOLUTION 3

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <limits.h>

// A structure to represent a stack
```

```

struct Stack
{
    unsigned capacity;
    int top;
    int *array;
};

// function to create a stack of given capacity.
struct Stack* createStack(unsigned capacity)
{
    struct Stack* stack = (struct Stack*) malloc(sizeof(struct Stack));
    stack -> capacity = capacity;
    stack -> top = -1;
    stack -> array =(int*) malloc(stack -> capacity * sizeof(int));
    return stack;
}

// Stack is full when top is equal to the last index
int isFull(struct Stack* stack)
{
    return (stack->top == stack->capacity - 1);
}

// Stack is empty when top is equal to -1
int isEmpty(struct Stack* stack)
{
    return (stack->top == -1);
}

// Function to add an item to stack. It increases
// top by 1
void push(struct Stack *stack, int item)
{
    if (isFull(stack))
        return;
    stack -> array[++stack -> top] = item;
}

// Function to remove an item from stack. It
// decreases top by 1
int pop(struct Stack* stack)
{
    if (isEmpty(stack))
        return INT_MIN;
    return stack -> array[stack -> top--];
}

//Function to show the movement of disks

```



```

void moveDisk(char fromPeg, char toPeg, int disk)
{
    printf("Move the disk %d from \'%c\' to \'%c\'\\n" disk, fromPeg, toPeg);
}

// Function to implement legal movement between
// two poles
void moveDisksBetweenTwoPoles(struct Stack *src, struct Stack *dest, char s, char d)
{
    int pole1TopDisk = pop(src);
    int pole2TopDisk = pop(dest);

    // When pole 1 is empty
    if (pole1TopDisk == INT_MIN)
    {
        push(src, pole2TopDisk);
        moveDisk(d, s, pole2TopDisk);
    }

    // When pole2 pole is empty
    else if (pole2TopDisk == INT_MIN)
    {
        push(dest, pole1TopDisk);
        moveDisk(s, d, pole1TopDisk);
    }

    // When top disk of pole1 > top disk of pole2
    else if (pole1TopDisk > pole2TopDisk)
    {
        push(src, pole1TopDisk);
        push(src, pole2TopDisk);
        moveDisk(d, s, pole2TopDisk);
    }
    // When top disk of pole1 < top disk of pole2
    else
    {
        push(dest, pole2TopDisk);
        push(dest, pole1TopDisk);
        moveDisk(s, d, pole1TopDisk);
    }
}

//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)
        moveDisksBetweenTwoPoles(src, dest, s, d);

    else if (i % 3 == 2)
        moveDisksBetweenTwoPoles(src, aux, s, a);

    else if (i % 3 == 0)
        moveDisksBetweenTwoPoles(aux, dest, a, d);
}
}

// Driver Program
int main()
{
    // Input: number of disks
    unsigned num_of_disks = 3;

    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;
}

```

SAMPLE OUTPUT:

Enter the number of discs:3

Movedisc1fromAtoC

Movedisc2fromAtoB

Movedisc1fromCtoB

Move disc3fromAtoC

Movedisc1fromBtoA

Movedisc2fromBtoC

Move disc1 from Ato C

TotalNumber ofmoves are:7

PROGRAM-6

6.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 sizeMAX)

- Insert anElement onto CircularQUEUE
- Delete an Element from CircularQUEUE
- Demonstrate *Overflow* and *Underflows* situations on Circular QUEUE
- Display the status of Circular QUEUE
- Exit

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

ALGORITHM:

Step 1:Start.

Step 2:Initialize queue size to MAX.

Step 3:Insert the elements into circular queue. If queue is full give a message as ‘queue is overflow’

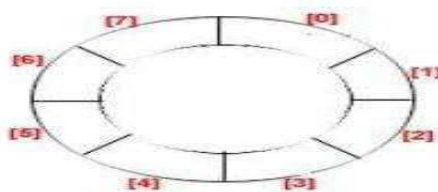
Step 4:Delete an element from the circular queue. If queue is empty give a message as ‘queue is underflow’.

Step 5:Display the contents of the queue.

Step 6:Stop.

ABOUT THE EXPERIMENT:

Circular queue is a linear data structure. It follows FIFO principle. In **circular queue** the last node is connected back to the first node to make a **circle**. **Circular** linked list follows the First In First Out principle. Elements are added at the rear end and the elements are deleted at front end of the **queue**. The queue is considered as a circular queue when the positions 0 and g MAX-1 are adjacent. Any position before front is also after rear. A circular queue looks like



Consider the example with Circular Queue implementation:

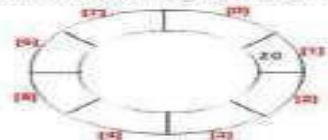
1) Initially: Front = 0 and rear = -1



2) Add item 10 then front = 0 and rear = 0.



3) Now delete one item then front = 1 and rear = 1.



4) Like this now insert 30, 40, and 50, 50, 70, 80 respectively then front = 1 and rear = 7.



5) Now in case of linear queue, we can not access 0 block for insertion but in circular queue next item will be inserted of 0 block then front = 0 and rear = 0.



SOLUTION 1

```
#include<stdio.h>
#include<stdlib.h>
#define max 10
int q[10],front=0,rear=-1; void
main()
{
int ch;
void insert();
void delet();
void display();
printf("\nCircular Queue operations\n");
printf("1.insert\n2.delete\n3.display\n4.exit\n"); while(1)
{
printf("Enter your choice:");
scanf("%d",&ch);
switch(ch)
{
case 1: insert(); break;
case 2: delet(); break;
case 3: display(); break;
case 4: exit(1);
default: printf("Invalid option\n");
}
}
}
void insert()
{
int x;
if((front==0&&rear==max-1)||((front>0&&rear==front-1))
printf("Queue is overflow\n");
else
{
printf("Enter element to be insert:"); scanf("%d",&x);
if(rear==max-1&&front>0)
{
rear=0;
q[rear]=x;
}
else
{
```

```
if((front==0&&rear== -1)|| (rear!=front-1))
q[++rear]=x;
}
}
}
void delet()
{
int a; if((front==0)&&(rear== -1))
{
printf("Queue is underflow\n"); exit(1);
}
if(front==rear)
{
a=q[front]; rear=-
1; front=0;
}
else if(front==max-1)
{
a=q[front];
front=0;
}
else a=q[front++];
printf("Deleted element is:%d\n",a);
}
void display()
{
int i,j; if(front==0&&rear== -1)
{
printf("Queue is underflow\n"); exit(1);
}
if(front>rear)
{
for(i=0;i<=rear;i++)
printf("\t%d",q[i]);
for(j=front;j<=max-1;j++)
printf("\t%d",q[j]);
printf("\nrear is at %d\n",rear);
printf("\nfront is at %d\n",front);
}
else
{
for(i=front;i<=rear;i++)
```

```
{  
printf("\t%d",q[i]);  
}  
printf("\nrear is at %d\n",rear);  
printf("\nfront is at %d\n",front);  
}  
printf("\n");  
}
```

Output:

Circular Queue operations 1.insert

2.delete

3.display

4.exit

Enter your choice:1

Enter element to be insert:1 Enter
your choice:1

Enter element to be insert:2 Enter
your choice:1

Enter element to be insert:3 Enter
your choice:1

Enter element to be insert:4 Enter
your choice:3

1 2 3 4 rear is at 4

front is at 1

Enter your choice:2 Deleted

element is:1 Enter your

choice:3 2 3 4

rear is at 4

front is at 2

Enter your choice:2 Deleted

element is:2 Enter your

choice:3

3 4 rear is at 4

front is at 3

Enter your choice:2
Deleted element is:3
Enter your choice:3 4
rear is at 4

front is at 4

Enter your choice:2
Deleted element is:4
Enter your choice:3
Queue is underflow

Process returned 1 (0x1) execution time : 28.719 s
Press any key to continue.

SOLUTION 2

```
#include<stdio.h>
#include<conio.h>
#define MAX 4
int ch, front=0,rear=1,count=0;
char q[MAX],item;
void insert()
{
    if(count==MAX)
        printf("\n Queue is Full");
    else
    {
        rear=(rear+1)%MAX;
        q[rear]=item;
    }
    count++;
}
void del()
{
    if(count==0)
        printf("\n Queue is Empty");
    else
        if(front>rear&&rear==MAX-1)
        {
            front=0;
            rear=-1;
            count=0;
        }
        else
        {
```



```
        item=q[front];
        printf("\n Deleted item is:%c",item);
        front=(front+1)%MAX;
        count--;
    }
}

void display()
{
    int i, f=front, r=rear;
    if(count==0)
        printf("\n Queue is Empty");
    else
    {
        printf("\n Contents of Queue is:\n");
        for(i=f;i<=r;i++)
        {
            printf("%c\t",q[i]);
            f=(f+1)%MAX;
        }
    }
}

void main()
{
    do
    {
        printf("\n 1:Insert\n 2:Delete\n 3:Display\n 4:Exit\n");
        printf("\n Enter the choice:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("\n Enter the character to be inserted:");
                    scanf("%c",&item);
                    insert();
                    break;
            case 2: del();
                    break;
            case 3: display();
                    break;
            case 4: exit(0);
                    break;
        } }while(ch!=4);
}
```

OUTPUT:

1.Insert 2. Delete 3. Display 4. Exit
Enter the choice: 1
Enter the character/ item to be inserted: A
1.Insert 2. Delete 3. Display4.Exit
Enter the choice: 1
Enter the character/ item to be inserted: B
1.Insert2. Delete 3. Display4. Exit
Enter the choice: 1
Enter the character/ item to be inserted: C
1.Insert 2. Delete 3. Display 4. Exit
Enter the choice: 1
Enter the character/ item to be inserted: D
1.Inser t2. Delete 3. Display4. Exit
Enter the choice: 3
Contents of Queue is:
ABCD

1.Insert 2. Delete 3. Display4. Exit
Enter the choice: 1
Enter the character/ item to be inserted: F
Queue is Full
1.Insert 2. Delete 3. Display 4. Exit
Enter the choice: 2
Deleted item is: A

PROGRAM-7

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

- Create a SLL of N Students Data by using *front insertion*.
- Display the status of SLL and count the number of nodes in it
- Perform Insertion and Deletion at End of SLL
- Perform Insertion and Deletion at Front of SLL
- Demonstrate how this SLL can be used as STACK and QUEUE
- Exit

ABOUT THE EXPERIMENT:

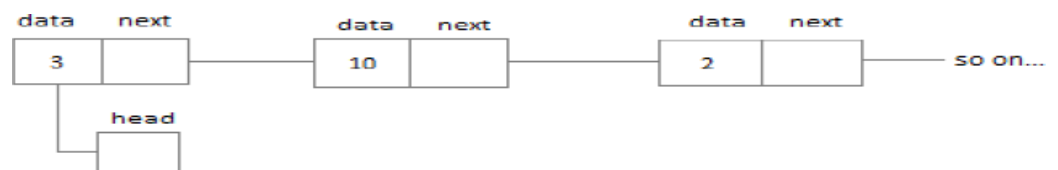
Linked List is a linear data structure and it is very common data structure which consists of group of nodes in a sequence which is divided in two parts.Each node consists of its own data and the address of the next node and forms a chain.Linked Lists are used to create trees and graphs.



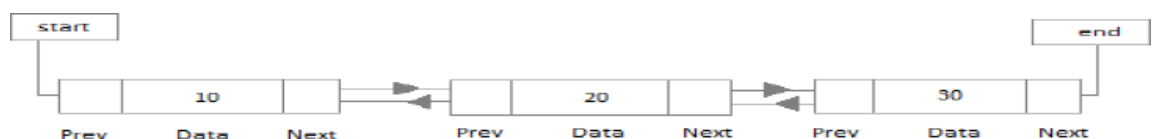
- They are dynamic in nature which allocates the memory when required.
- Insertion and deletion operations can be easily implemented.
- Stacks and queues can be easily executed.
- Linked List reduces the access time.
- Linked lists are used to implement stacks, queues, graphs, etc.
- Linked lists let you insert elements at the beginning and end of the list.
- In Linked Lists we don't need to know the size in advance.

Types of Linked List:

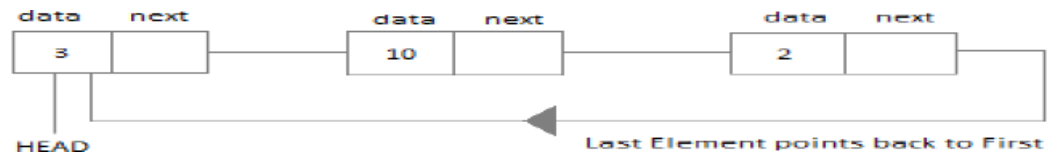
Singly Linked List: Singly linked lists contain nodes which have a data part as well as an address part i.e. next, which points to the next node in sequence of nodes. The operations we can perform on singly linked lists are insertion, deletion and traversal.



Doubly Linked List: In a doubly linked list, each node contains two links the first link points to the previous node and the next link points to the next node in the sequence.



- **Circular Linked List:** In the circular linked list the last node of the list contains the address of the first node and forms a circular chain.



ALGORITHM:

Step 1: Start.
 Step 2: Read the value of N
 Step 3: Display
 Step 4: Count
 Step 5: Perform insertion at front of list.
 Step 6: Perform deletion at front of the list.
 Step 7: Perform insertion at end of the list.
 Step 8: Perform deletion at the end of the list.
 Step 9: show singly linked list usage as stack
 Step 10: show singly linked list usage as queue.
 Step 11: Stop

SOLUTION 1

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h> int
count=0;
struct node
{
int sem,phno;
char name[20],branch[10],usn[20]; struct
node *next;
}*first=NULL,*last=NULL,*temp=NULL, *temp1;
void create()
{
int sem,phno;
char name[20],branch[10],usn[20];
temp=(struct node*)malloc(sizeof(struct node));
printf("\n Enter usn,name, branch, sem, phno of student : ");
scanf("%s %s %s %d %d", usn, name,branch, &sem,&phno);
strcpy(temp->usn,usn);
strcpy(temp->name,name);
strcpy(temp->branch,branch); temp-
>sem = sem;
  
```

```

temp->phno = phno;
temp->next=NULL;
count++;
}
void insert_atfirst()
{
if (first == NULL)
{
create(); first =
temp; last = first;
}
else
{
create();
temp->next = first; first =
temp;
}
}
void insert_atlast()
{
if(first==NULL)
{
create(); first =
temp; last = first;
}
else
{
create();
last->next = temp;
last = temp;
}
}
void display()
{
temp1=first; if(temp1 ==
NULL)
{
printf("List empty to display \n");
return;
}
printf("\n Linked list elements from Beginning : \n"); while
(temp1!= NULL)
printf("%s %s %s %d %d\n", temp1->usn, temp1->name,temp1->branch,temp1->sem,temp1->phno );
temp1 = temp1->next;

```

```

}
printf(" No of students = %d ", count);
}
int deleteend()
{
    struct node *temp; temp=first;
    if(temp->next==NULL)
    {
        free(temp);
        first=NULL;
    }
    else
    {
        while(temp->next!=last)
        temp=temp->next;
        printf("%s %s %s %d %d\n", last->usn, last->name,last->branch, last->sem, last->phno );
        free(last);
        temp->next=NULL;
        last=temp;
    }
    count--; return
    0;
}
int deletefront()
{
    struct node *temp; temp=first;
    if(temp->next==NULL)
    {
        free(temp);
        first=NULL; return
        0;
    }
    else
    {
        first=temp->next;
        printf("%s %s %s %d %d", temp->usn, temp->name,temp->branch,temp->sem, temp->phno );
        free(temp);
    }
    count--;
    return 0;
}
void main()
{

```

```

int ch,n,i;
first=NULL;
temp = temp1 = NULL;
printf(".....MENU.....\n");
printf("\n 1 - create a SINGLE LINKED LIST of n Students");
printf("\n 2 - Display from Beginning");
printf("\n 3 - Insert at end"); printf("\n 4 -
delete at end"); printf("\n 5 - Insert at
Beginning"); printf("\n 6 - delete at
Beginning"); printf("\n 7 - exit\n");
printf(".....\n"); while
(1)
{
printf("\n Enter choice : ");
scanf("%d", &ch);
switch (ch)
{
case 1: printf("\n Enter no of students : "); scanf("%d",
&n);
for(i=0;i<n;i++)
insert_atfirst(); break;
case 2: display(); break;
case 3: insert_atlast();
break;
case 4: deleteend(); break;
case 5: insert_atfirst();
break;
case 6: deletefront();
break;
case 7: exit(0);
default: printf("wrong choice\n");
}
}
}

```

Output:

```

.....MENU.....

1 - create a SINGLE LINKED LIST of n emp 2 -
Display from Beginning
3 - Insert at end
4 - delete at end

```

5 - Insert at Beginning
6 - delete at Beginning
7 - exit

Enter choice : 1

Enter no of students : 1

Enter usn,name, branch, sem, phno of student :
1VE15IS001 GURU CSE 6 28566200

Enter choice : 2

Linked list elements from Beginning :
1VE15IS001 GURU CSE 6 28566200 No of students = 1

Enter choice : 3

Enter usn,name, branch, sem, phno of student : 1VE15IS002 PRASAD CSE 6 28555200
Enter choice : 2

Linked list elements from Beginning :
1VE15IS001 GURU CSE 6 28566200
1VE15IS002 PRASAD CSE 6 28555200 No
of students = 2
Enter choice : 4
1VE15IS002 PRASAD CSE 6 28555200

Enter choice : 2

Linked list elements from Beginning :
1VE15IS001 GURU CSE 6 28566200 No of students = 1
Enter choice : 5

Enter usn,name, branch, sem, phno of student :
1VE15IS003 GURUPRASAD CSE 6 2855 0000

Enter choice : 2

Linked list elements from Beginning : 1VE15IS003
GURUPRASAD CSE 6 28550000 1VE15IS001 GURU
CSE 6 28566200
No of students = 2

Enter choice : 6
1VE15IS003 GURUPRASAD CSE 6 28550000

Enter choice : 2

Linked list elements from Beginning : 1VE15IS001
GURU CSE 6 28566200
No of students = 1
Enter choice : 7

Process returned 0 (0x0) execution time : 117.734 s
Press any key to continue.

SOLUTION 2

```
#include<stdio.h>
#include<conio.h>
int MAX=4,count;
struct student
{
    char usn[10];
    char name[30];
    char branch[5];
    int sem;
    char phno[10];
    struct student *next; //Self referential pointer.
};

typedef struct student NODE;

int count nodes(NODE*head)
{
    NODE*p;
    count=0;
    p=head;
    while(p!=NULL)
    {
        p=p->next;
        count++;
    }
    return count;
}

NODE * getnode(NODE*head)
{
    NODE *newnode;
    Newnode=(NODE*)malloc(sizeof(NODE));
```

```

//Create firstNODEprintf("\nEnterUSN, Name, Branch,Sem,Ph.No\n");
flushall();
gets(newnode->usn);
flushall();
gets(newnode->name);
flushall();
gets(newnode->branch);
scanf("%d",&(newnode->sem));
flushall();
gets(newnode->phno);
newnode->next=NULL;    //Set next to NULL...head=newnode;
returnhead;
}
NODE* display(NODE*head)
{
    NODE*p;
    if(head == NULL)
        printf("\nNostudentdata\n");
    else
    {
        p=head;
        printf("\n----STUDENTDATA---\n");
        printf("\nUSN\tNAME\tBRANCH\tSEM\tPh.NO.");
        while(p!=NULL)
        {
            printf("\n%s\t%s\t%s\t%d\t%s", p->usn, p->name, p->branch, p->sem,p->phno);
            p =p->next;    //Go to next node...
        }
        printf("\nThe no. ofnodes in listis: %d",countnodes(head));
        return head;
    }
}

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

NODE* insert_front(NODE*head)
{

```

```

        if(countnodes(head)==MAX)
            printf("\nList isFull / Overflow!!");
        else
            head=create(head);//create()insertnodesatfront.
        returnhead;
    }
}

NODE* insert_rear(NODE*head)
{
    NODE*p,*newnode;
    p=head;
    if(countnodes(head)==MAX)
        printf("\nList isFull(Queue)!!");
    else
    {
        if(head == NULL)
        {
        }
        else
        {
            Newnode=getnode(head);
            head=newnode;        //set first node to be head
            newnode=getnode(head);
            while(p->next!=NULL)
            {
                p=p->next;
            }
            p->next=newnode;
        }
    }
    returnhead;
}

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

```

```
}
```

```
NODE* delete_front(NODE *head)
{
    NODE*p;
    if(head==NULL)
        printf("\nList is Empty/Underflow(STACK/QUEUE)");
    else
    {
    }
    p=head;
    head=head->next;    //Set 2nd NODE as headfree(p);
    printf("\nFront(first) node is deleted");
    return head;
}

NODE* delete_rear(NODE *head)
{
    NODE*p, *q;
    p=head;
    while(p->next!=NULL)
    {
        p=p->next; //Go upto -1 NODE which you want to delete
    }
    q=p->next;
    free(q); //Delete last NODE...p->next=NULL;
    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_rear(head); break;
            case 3: break;
        }
        head=display(head);
    } while(ch!=3); return head;
}

NODE* stack(NODE*head)
{
    int ch;
    do
```

```

    {
        printf("\nSSLused as Stack...");
        printf("\n1.Insertat Front(PUSH)\t 2.Delete fromFront(POP)\t3.Exit");
        printf("\nEnter your choice:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: head=insert_front(head);
            break;
            case 2: head=delete_front(head);
            break;
            case 3: break;
        }
        head=display(head);
    } while(ch!=3);return head;}
NODE* queue(NODE *head)
{
    int ch;do
    printf("\nSSLused as Queue...");
    printf("\n1.InsertatRear(INSERT)\t 2.Deletefrom Front(DELETE)\t3.Exit");
    printf("\nEnter your choice:");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1: head=insert_rear(head);
        break;
        case 2: head=delete_front(head);
        break;
        case 3: break;
    }
    head=display(head);
} while(ch!=3);return head;
}

void main()
{
    int ch, i, n;NODE*head;
    head=NULL;
    clrscr();
    printf("\n*-----StudedntDatabase----- *");
    do
    {
        printf("\n1.Create\t2.Display\t3.Insert\t4.Delete\t5.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++)

```

```

        head=create(head);//Call to Create NODE...
        break;
        case 2: head=display(head); //Call to Display...break;
        case 3: head=insert(head); //Call to Insert...break;
        case 4: head=del(head); //Call to deletebreak;
        case 5: head=stack(head);break;
        case 6: head=queue(head);break;
        case 7: exit(); //Exit...
    }
}while(ch!=7);
}

```

OUTPUT:

1. Create 2. Display3.Insert 4. Delete 5. Stack 6.Queue7. Exit

Enteryourchoice: 1

How many student data you want to create:2

EnterUSN,Name,Branch,Sem, Ph.No

1kt12cs001 kumar cs 3 9900099000

EnterUSN,Name,Branch,Sem, Ph.No

1kt12is002 ravi is 3 9900099111

1. Create 2. Display3.Insert 4. Delete 5. Stack 6.Queue7. Exit

Enteryourchoice: 2

----STUDENTDATA----

USN	NAME	BRANCH	SEM	Ph.NO.
1kt12is002	ravi	is	3	9900099111
1kt12cs001	kumar	cs	3	9900099000

The no. of nodes in list is: 2

1. Create 2. Display 3.Insert 4. Delete 5. Stack 6.Queue 7. Exit

Enteryourchoice: 3

1.Insert at Front(First) 2.Insert at End(Rear/Last) 3.Exit

Enteryourchoice: 1

EnterUSN,Name,Branch,Sem, Ph.No

1kt12cs003 suresh cs 3 99000992222

----STUDENTDATA----

USN	NAME	BRANCH	SEM	Ph.NO.
1kt12cs003	suresh	cs	3	99000992222
1kt12is002	ravi	is	3	9900099111
1kt12cs001	kumar	cs	3	9900099000

The no. of nodes in list is: 3

1.InsertatFront(First)2.InsertatEnd(Rear/Last)3.ExitEnteryourchoice: 2

EnterUSN,Name,Branch,Sem, Ph.No

1kt12is004 naresh is 3 99000993333

----STUDENTDATA----

USN

1kt12cs003 NAME

suresh BRANCH

cs SEM

3 Ph.NO.

99000992222

1kt12is002 ravi is 3 9900099111

1kt12cs001 kumar cs 3 9900099000

1kt12is004 naresh is 3 99000993333

Theno. of nodes in list is: 4

1.Insert atFront(First)2.InsertatEnd(Rear/Last)3.Exit

Enteryourchoice: 3

1. Create 2. Display3.Insert 4. Delete 5. Stack 6.Queue7. Exit

2. Enteryourchoice: 4

1. Delete from Front (First) 2. Delete from End (Rear/Last) 3.Exit

2. Enteryourchoice: 1

Front(first) nodeis deleted

----STUDENTDATA----

USN

1kt12is002 NAME

ravi BRANCH

is SEM

3 Ph.NO.

9900099111

1kt12cs001 kumar cs 3 9900099000

1kt12is004 naresh is 3 99000993333

Theno. of nodes in list is: 3

1. Delete from Front (First) 2. Delete from End (Rear/Last) 3.Exit

2. Enteryourchoice: 2

Last (end) nodeis deleted

----STUDENTDATA----

USN NAMEBRANCH SEM Ph.NO.

1kt12is002 ravi is 3 9900099111

1kt12cs001 kumar cs 3 9900099000Theno. of nodes in list is: 2

1. Delete from Front (First) 2. Delete from End (Rear/Last) 3.Exit

2. Enteryourchoice: 3

1. Create 2. Display3.Insert 4. Delete 5. Stack 6.Queue7. Exit

2. Enteryourchoice: 5

SSL used as Stack...

1.Insert at Front(PUSH)2.Delete from Front(POP))3.Exit

Enter your choice: 1

Enter USN,Name,Branch,Sem, Ph.No : 1kt12is010 vikkyis 3 9900011111

----STUDENTDATA----

USN NAMEBRANCH SEM Ph.NO.

1kt12is010	vikky	is	3	9900011111
1kt12is002	ravi	is	3	9900099111
1kt12cs001	kumar	cs	3	9900099000

1.Create 2. Display 3.Insert 4. Delete 5. Stack 6.Queue7. Exit
Enter your choice: 7

PROGRAM-8

8. Design, develop and Implement a menu driven Program in C for the following operations on Doubly Linked List(DLL) of Employee Data with the fields:

SSN, Name, Dept, Designation, Sal, PhNo.

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

ABOUT THE EXPERIMENT:

Doubly Linked List: In a doubly linked list, each node contains two links the first link points to the previous node and the next link points to the next node in the sequence. In computer science, a **doubly linked list** is a linked data structure that consists of a set of sequentially linked records called nodes. Each node contains two fields, called *links*, that are references to the previous and to the next node in the Sequence of nodes. The beginning and ending nodes' **previous** and **next** links, respectively.

A Doubly linked list whose nodes contain three fields: an integer value, the link to the next node, and the link to the previous node. The two node links allow traversal of the list in either direction. While adding or removing a node in a doubly linked list requires changing more links than the same operations on a singly linked list, the operations are simple and potentially more efficient (for nodes other than first nodes) because there is no need to keep track of the previous node during traversal or no need to traverse the list to find the previous node, so that its link can be modified.

ALGORITHM:

- Step 1: Start.
- Step 2: Read the value of N. (N student's information)
- Step 3: Create a doubly linked list. (DLL)
- Step 4: Display the status of DLL.
- Step 5: Count the number of nodes.
- Step 6: Perform insertion at front of list.
- Step 7: Perform deletion at the front of the list.
- Step 8: Perform insertion at end of the list.
- Step 9: Perform deletion at the end of the list.
- Step 10: Demonstrate how doubly linked list can be used as double ended queue.
- Step 11: Stop.

Solution-1

```
#include<stdio.h>
#include<conio.h>
int MAX=4, count;
struct emp
{
    int ssn;
    char name[20];
```

```

        chardept[10];
        chardesig[15];
        int sal;
        char phno[10];
        struct emp *left;struct emp *right;
    };

typedef struct empNODE;

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);
    fflush();
    gets(newnode->name);
    fflush();
    gets(newnode->dept);
    fflush();
    gets(newnode->desig);
    scanf("%d",&newnode->sal);
    fflush();
    gets(newnode->phno);
    head=newnode;
    return head;
}

NODE* display(NODE*head)
{
    NODE*p;
    if(head==NULL)
        printf("\nNoEmployee data\n");
    else
    {
        p=head;
        printf("\n----EMPLOYEE DATA--- \n");
    }
}

```

```

        printf("\nSSN\tNAME\tDEPT\tDESIGNATION\tSAL\tPh.NO.");
        while(p!=NULL)
        {
            p->sal,p->phno);
        }
        printf("\n%d\t%s\t%s\t%s\t%d\t%s",p->ssn,p->name,p->dept,p->desig,p = p->right;

        printf("\nThe no. Of nodes in list is: %d",count nodes(head))
    }
    Return head;
}

NODE* create(NODE*head)// creating & inserting at end.
{
    NODE*p,*newnode;
    p=head;
    if(head==NULL)
    {
    }
    else
    {
        Newnode=getnode(head);
        head=newnode;
        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 isFull!!");
    else
        head=create(head);
    return head;
}

NODE* insert_front(NODE*head)
{
    NODE*p,*newnode;
    if(countnodes(head)==MAX)
        printf("\nList isFull!!");
    else
    {
        if(head==NULL)

```

```

        {
            Newnode=getnode(head);
            head=newnode;        //set first node to be head
        }
        else
        {
        }
    }

    Newnode=getnode(head);
    Newnode->right=head;
    head->left=newnode;
    head=newnode;
    return head;
}

NODE* insert(NODE *head)
{
    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: 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;    //Go upto -1 node whichyouwant to delete
    }
    q=p->left;
    q->right=NULL;
    p->left=NULL;
    free(p); //Delete last node...printf("\nLast(end) entryis deleted");returnhead;
}
NODE*del(NODE *head)
{
    int ch;
    do
    {
        printf("\n1.Delete fromFront(First)\t2. Deletefrom 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("\nDLLUsed asDouble EndedQueue");
    printf("\n1.QUEUE-InsertatRear&DeletefromFront");
    printf("\n2.QUEUE-InsertatFront&Delete fromRear");
    printf("\n3.Exit");
    printf("\nEnter your choice:");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:
            do{
                printf("\n1.Insertat Rear\t2.Delete fromFromFront\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.InsertatFront\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(ch!=3);
        }while(ch2!=3);
        break;
        case 3: break;
        head=display(head);return head;
}
void main()
{
        int ch, i, n; NODE*head;
        head=NULL;
        clrscr();
        printf("\n-----EmployeeDatabase----- ");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);//Call to Create node...
                break;
        case 2: head=display(head); //Call to Display...
                break;
        case 3: head=insert(head); //Call to Insert...
                break;
        case 4: head=del(head); //Call to delete
                break;
        case 5: head=queue(head);
                break;
        }
        }
}

```

```

        case 6: exit(0); //Exit...
            break;
    }
} while(ch!=6);
}

```

OUTPUT:

-----EmployeeDatabase-----

1.Create 2. Display 3.Insert 4. Delete 5. Queue 7. Exit

Enter your choice: 1

How many employees data you want to create: 2

Enter SSN, Name,Dept,Designation,Sal,Ph.No

1 KUMAR CSE INSTRUCTOR 8000 900099000

Enter SSN, Name,Dept,Designation,Sal,Ph.No

2 RAVI ISE INSTRUCTOR 9000 900099111

1.Create 2. Display 3.Insert 4. Delete 5. Queue 7. Exit

Enter your choice: 2

----EMPLOYEE DATA----

SSN	NAME	DEPT	DESIGNATION	SAL	Ph.NO.
1	KUMAR	CSE	INSTRUCTOR	8000	900099000
2	RAVI ISE	INSTRUCTOR	9000	900099111	

The no. of nodes in list is: 2

1. Create 2. Display 3.Insert 4. Delete 5. 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 SSN, Name,Dept,Designation,Sal,Ph.No

3 JIM CSE ATTENDER 6000 900099333

----EMPLOYEE DATA----

SSN	NAME	DEPT	DESIGNATION	SAL	Ph.NO.
3	JIM	CSE	ATTENDER	6000	900099333
1	KUMAR	CSE	INSTRUCTOR	8000	900099000
2	JIM	CSE	INSTRUCTOR	9000	900099111

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

1. Create 2. Display 3.Insert 4. Delete 5. Queue 7. Exit

Enter your choice: 5

DLL used as Double Ended Queue

1. QUEUE-Insert at Rear & Delete from Front 2. QUEUE-Insert at Front & Delete from Rear 3. Exit

Enter your choice: 3

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

Solution-2

```
#include<string.h>

int count=0;
struct node
{
    struct node *prev; int
    ssn,phno;
    float sal;
    char name[20],dept[10],desg[20];
    struct node *next;
} *h,*temp,*temp1,*temp2,*temp4; void
create()
{
    int ssn,phno; float
    sal;
    char name[20],dept[10],desg[20];
    temp =(struct node *)malloc(sizeof(struct node)); temp-
    >prev = NULL;
    temp->next = NULL;
    printf("\n Enter ssn,name,department, designation, salary and phno of employee : ");
    scanf("%d %s %s %s %f %d", &ssn, name,dept,desg,&sal, &phno);
    temp->ssn = ssn; strcpy(temp-
    >name,name); strcpy(temp-
    >dept,dept); strcpy(temp-
    >desg,desg); temp->sal = sal;
    temp->phno = phno;

    count++;
}
void insertbeg()
{
    if (h == NULL)
    {
        create();
        h = temp; temp1 =
        h;
    }
}
```



```

else
{
create();
temp->next = h;
h->prev = temp;
h = temp;
}
}
void insertend()
{
if(h==NULL)
{
create();
h = temp; temp1 =
h;
}
else
{
create();
temp1->next = temp; temp-
>prev = temp1; temp1 = temp;
}
}
void displaybeg()
{
temp2 =h;
if(temp2 == NULL)
{
printf("List empty to display \n");
return;
}
printf("\n Linked list elements from begining : \n");
while (temp2!= NULL)
{
printf("%d %s %s %s %f %d\n", temp2->ssn, temp2->name,temp2->dept,temp2->desg,temp2->sal,
temp2->phno );
temp2 = temp2->next;
}
printf(" No of employees = %d ", count);
}
int deleteend()
{

```

```

struct node *temp; temp=h;
if(temp->next==NULL)
{
free(temp);
h=NULL; return
0;
} else
{
temp2=temp1->prev; temp2-
>next=NULL;
printf("%d %s %s %s %f %d\n", temp1->ssn, temp1->name,temp1->dept,
temp1->desg,temp1->sal, temp1->phno );
free(temp1);
}
count--;
return 0;
}
int deletebeg()
{
struct node *temp; temp=h;
if(temp->next==NULL)
{
free(temp);
h=NULL;

}
else
{
h=h->next;
printf("%d %s %s %s %f %d", temp->ssn, temp->name,temp->dept,
temp->desg,temp->sal, temp->phno );
free(temp);
}
count--;
return 0;
}
void main()
{
int ch,n,i;
h=NULL;
temp = temp1 = NULL;
printf(".....MENU.....\n");
printf("\n 1 - create a DLL of n emp");

```

```

printf("\n 2 - Display from beginning");
printf("\n 3 - Insert at end");
printf("\n 4 - delete at end");
printf("\n 5 - Insert at beg");
printf("\n 6 - delete at beg");
printf("\n 7 - exit\n");
printf(".....\n");
while (1)
{
printf("\n Enter choice : ");
scanf("%d", &ch);
switch (ch)
{
case 1:
printf("\n Enter no of employees : ");
scanf("%d", &n);
for(i=0;i<n;i++)
insertend();
break;
case 2:
displaybeg();
break;
case 3:
insertend();
break;
case 4:
deleteend();
break;
case 5:
insertbeg();
break;
case 6:
deletebeg();
break;
case 7:
exit(0);
default:printf("wrong choice");
}
}
}

```

OUTPUT:

.....MENU.....

- 1 - create a DLL of n emp
- 2 - Display from beginning

3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit

Enter choice : 1

Enter no of employees : 1

Enter ssn,name,department,
designation, salary and phno
of employee : 01 ABC IS E
AP 35000 28566220

Enter choice : 2
Linked list elements from begining :
Enter choice : 3

Enter
ssn,name,department,
designation, salary and
phno of employee : 02 PQR
IS E AP 30000 28555220

Enter choice : 2

Linked list elements from begining :
1 ABC CSE AP 35000.000000 28566220
Enter choice : 4
2 PQR CSE AP 30000.000000 28555220

Enter choice : 5

Enter ssn,name,department,
designation, salary and phno
of employee : 03 XYZ IS E
AP 40000 28444220

Enter choice : 7

Process returned 0 (0x0) execution time :
136.750 s
Press any key to continue.

Solution -3

```
#include<string.h>
int count=0;
struct node
{
    struct node *prev;int ssn,phno;
    float sal;
    char name[20],dept[10],desg[20];
    struct node *next;
} *h,*temp,*temp1,*temp2,*temp4;

void create()
{
    int ssn,phno;
    float sal;
    char name[20],dept[10],desg[20];
    temp =(struct node *)malloc(sizeof(struct node));
    temp->prev = NULL;
    temp->next = NULL;
    printf("\n Enter ssn,name,department, designation, salary and phno of employee : ");
    scanf("%d %s %s %s %f %d", &ssn, name,dept,desg,&sal, &phno);
    temp->ssn = ssn;
    strcpy(temp->name,name);strcpy(temp->dept,dept);
    strcpy(temp->desg,desg);
    temp->sal = sal;
    temp->phno = phno;
    count++;
}
void insertbeg()
{
    if (h == NULL)
    { create(); h = temp;
      temp1 = h;
    }
    else
    { create();
      temp->next = h;
      h->prev = temp;
      h = temp;
    }
}
void insertend()
{
    if(h==NULL)
    { create(); h = temp;
      temp1 = h;
    }
}
```

```
}
else
{
create();
temp1->next = temp;
temp->prev = temp1;
temp1 = temp;
}
}
void displaybeg()
{
temp2 =h;
if(temp2 == NULL)
{
printf("List empty to display \n");
return;
}
printf("\n Linked list elements from begining : \n");
while (temp2!= NULL)
{
printf("%d %s %s %s %f %d\n", temp2->ssn, temp2->name,temp2->dept,temp2->desg,temp2->sal,
temp2->phno );
temp2 = temp2->next;
}
printf(" No of employees = %d ", count);
}
int deleteend()
{
struct node *temp;temp=h;
if(temp->next==NULL)
{ free(temp);h=NULL;
return 0;
}
else
{
temp2=temp1->prev;temp2->next=NULL;
printf("%d %s %s %s %f %d\n", temp1->ssn, temp1->name,temp1->dept,temp1->desg,temp1->sal,
temp1->phno );
free(temp1);}
count--; r
eturn 0;
}
int deletebeg()
{
struct node *temp;temp=h;
if(temp->next==NULL)
{ free(temp);h=NULL;
}
}
```

```

h=h->next;
printf("%d %s %s %s %f %d", temp->ssn, temp->name,temp->dept,temp->desg,temp->sal, temp-
>phno );
free(temp);
}
count--;
return 0;
}
void main()
{
int ch,n,i;
h=NULL;
temp = temp1 = NULL;
printf("MENU\n");
printf("\n 1 - create a DLL of n emp");
printf("\n 2 - Display from beginning");printf("\n 3 - Insert at end");
printf("\n 4 - delete at end");printf("\n 5 - Insert at beg");printf("\n 6 - delete at beg");
printf("\n 7 - exit\n");
printf("\n");
while (1)
{
printf("\n Enter choice : ");
scanf("%d", &ch);
switch (ch)
{
case 1: printf("\n Enter no of employees : ");scanf("%d", &n);
for(i=0;i<n;i++)
insertend();
break;
case 2:
displaybeg();
break;
case 3:
insertend();
break;
case 4:
deleteend();
break;
case 5:
insertbeg();
break;
case 6:
deletebeg();
break;
case 7: exit(0);default:
printf("incorrect choice");
}
}
}

```

MENU.....
–Create a DLL of n emp
Display from beginning
Insert at end

Delete at end
Insert at beg
Delete at beg
exit

Enter choice : 1
Enter no of employees : 2
Enter ssn,name,department, designation, salary and phno of employee :1RAJ SALES MANAGER
15000 911
Enter ssn,name,department, designation, salary and phno of employee :2
RAVI HR ASST 10000 123
Enter choice : 2
Linked list elements from begining :
1 RAJ SALES MANAGER 15000.000000 911
2 RAVI HR ASST 10000.000000 123 No
of employees = 2Enter choice : 3
Enter ssn,name,department, designation, salary and phno of employee : 3RAM MARKET MANAGER
50000 111

Enter choice : 2
Linked list elements from begining :
1 RAJ SALES MANAGER 15000.000000 911
2 RAVI HR ASST 10000.000000 123
3 RAM MARKET MANAGER 50000.000000
111 No of employees = 3Enter choice : 4
3 RAM MARKET MANAGER 50000.000000
111 Enter choice : 2
Linked list elements from begining :
1 RAJ SALES MANAGER 15000.000000 911
2 RAVI HR ASST 10000.000000
123 No of employees = 2

Enter choice : 5
Enter ssn,name,department, designation, salary and phno of employee :0 ALEX EXE TRAINEE 2000
133
Enter choice : 2
Linked list elements from begining :
ALEX EXE TRAINEE 2000.000000 133
RAJ SALES MANAGER 15000.000000 911
2 RAVI HR ASST 10000.000000
123 No of employees = 3

Enter choice : 6
ALEX EXE TRAINEE 2000.000000 133

Enter choice : 2
Linked list elements from begining :
RAJ SALES MANAGER 15000.000000 911
2 RAVI HR ASST 10000.000000 123

No of employees = 2Enter choice : 7
Exit

/*9Q. Design, Develop and Implement a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes

- Represent and Evaluate a Polynomial $P(x,y,z)=6x^2y^2z-4yz^5+3x^3yz+2xy^5z-2xyz^3$**
- Find the sum of two polynomials $POLY1(x,y,z)$ and $POLY2(x,y,z)$ and store the result in $POLYSUM(x,y,z)$**

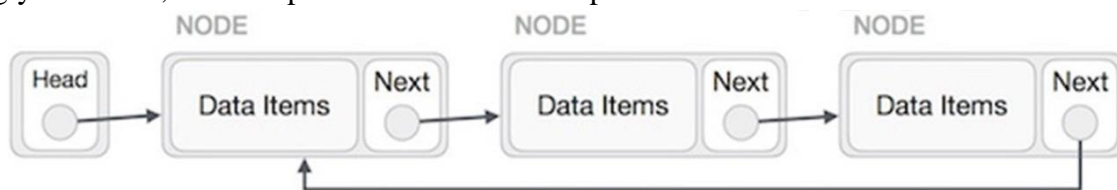
Support the program with appropriate functions for each of the above operations.*/

Theory:

Circular Linked List is a variation of Linked list in which the first element points to the last element and the last element points to the first element. Both Singly Linked List and Doubly Linked List can be made into a circular linked list.

Singly Linked List as Circular

In singly linkedlist, the next pointer of the last node points to the first node.



Let's find the sum of the following two polynomials

$$(3y^5 - 2y + y^4 + 2y^3 + 5) \text{ and } (2y^5 + 3y^3 + 2 + 7)$$

1) Write in Standard form $(3y^5 + y^4 + 2y^3 - 2y + 5) + (2y^5 + 3y^3 + 7y + 2)$

2) Arrange in columns of like terms and then add

$$\begin{array}{r}
 3y^5 + y^4 + 2y^3 - 2y + 5 \\
 2y^5 \quad \quad + 3y^3 + 7y + 2 \\
 \hline
 5y^5 + y^4 + 5y^3 + 5y + 7
 \end{array}$$

© mathwarehouse.com

ALGORITHM:

Algorithm: Polynomial_using_SCLL

- Start
- Read choice for polynomial operation
- Switch choice

1: readpolynomial1a=readpoly()

2: readpolynomial2b=readpoly()

3: print polynomial print(a)

4: printpolynomial2print(b)

5: addpolynomial1andpolynomial2 Andprintanswerc=polyadd(a,b)Print(c)

6: evaluatepolynomialCal)

7: exit

Algorithm: getnode()

Start Create newnode and
return it Stop

Algorithm:

```

    readpoly()
    1.Start
    2.Read coefficient and exponents
    3.Askforcombinationifyesrepeat step2
    4.stop

```

Algorithm:compare(a,b)

```

    1.Start
        ifa->x>b->xreturn1
        Else
        if a->y>b->yreturn1
        else
        ifa->z>b->zreturn1
        else
        ifa->x<b->xreturn-1
        else
        ifa->y<b->yreturn-1
        else
        ifa->z<b->zreturn-1
    2.Return 0
    3.stop

```

Algorithm:attach(ef,x1,y1,z1,ptr)

```

    1.start
    2.attachtoptr
    3.stop

```

Algorithm:cpadd(a,b)

```

    1.start
    2.compare(a-1:attach(b->coeff,b->x,b->y,b->z,lastc)0:sum=a->coeff+b->coeff
    attach(sum,a->x,a->y,a->z,lastc)1.attach(a->coeff,a->x,a->y,a->z,lastc)
    3.repeatstep2untilallcoefficientfroma,bprocessed4.stop

```

Algorithm:print(ptr)

```

    1.start
    2.displaycoefficientandexponents3.repeatstep2untilcur->link!=ptr
    3.stop

```

Algorithm:Cal(ptr)

```

    1.start
    2.readvaluesforx,yandz
    3.substitutevalueinpolynomialcalculateand
    4.stop

```

SOLUTION-1

```

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
structnode//polynomial node
{
    Int coef;intx,y,z;
    Struct node*link;

```

```

};
typedef struct node*NODE;

NODE get node()//create anode
{
    NODEx;
    x=(NODE)malloc(sizeof(structnode));
    return x;
} //end of getnode

NODE readpoly()
{
    NODE temp,head,cur;
    char ch;
    head=getnode();//create a head node and set all values to -1 it is similar to
    FIRST in SLL program
    head->coef=-1;head->x=-1;head->y=-1;head->z=-1;
    head->link=head;//self-reference
    do
    {
        temp=getnode();//create a polynomial node
        printf("\nEnter the coefficient and exponent in decreasing order");
        scanf("%d%d%d%d",&temp->coef,&temp->x,&temp->y,&temp->z);
        cur=head;
        while(cur->link!=head)//find the last node
        cur=cur->link;
        cur->link=temp;//connect new node to the last node
        temp->link=head;//point back to head
        printf("\nDo you want to enter more coefficients(y/n)");
        fpurge(stdin);//to clear the stdin buffer
        scanf("%c",&ch);
    } while(ch=='y' || ch=='Y');

    Return head;//return the polynomial list
} //end of readpoly

Int compare(NODE a,NODE b)//function to compare the A and B polynomial nodes
{
    if(a->x>b->x)
        return 1;
    else
        if(a->x<b->x)
            return -1;
        elseif(a->y>b->y)
            return 1;
        else
            if(a->y<b->y)
                return -1;
            else

```

```

        if(a->z>b->z)
            return 1;
        else
            if(a->z<b->z)
                return -1;
            return 0;
    } //end of compare

```

Void attach(int cf,int x1,int y1,int z1,NODE *ptr)//function to attach the A and B polynomial node to C Polynomial

```

{
    NODE temp;
    temp=getnode();
    temp->coef=cf;
    temp->x=x1;
    temp->y=y1;
    temp->z=z1;
    (*ptr)->link=temp;
    *ptr=temp;
} //end of attach

```

NODE add poly(NODEa,NODEb)//function to add polynomial A and B i.e, C=A+B

```

{
    NODE start a,c,last c;
    Int sum,done=0;
    Start a=a;
    a=a->link;
    b=b->link;
    c=getnode();//create list C to store A+B
    c->coef=-1;
    c->x=-1;
    c->y=-1;
    c->z=-1;
    lastc=c;
    do{
        switch(compare(a,b))
        {
            Case-1:attach(b->coef,b->x,b->y,b->z,&lastc);
            b=b->link;
            break;
            case 0:if(starta==a)
                done=1;
            else{
                sum=a->coef+b->coef;
                if(sum)attach(sum,a->x,a->y,a->z,&lastc);
                a=a->link;b=b->link;
            }
        }
    }
}

```

```

    }
    break;
    case 1: if(starta==a) done=1;
    attach(a->coef, a->x, a->y, a->z, &lastc);
    a=a->link;
    break;
}
}
while(!done); // repeat until not done
last c->link=c; // point back to head of C
return c; // return answer
}

```

Void print(NODE ptr) // to print the polynomial

```

{
    NODE cur;
    cur=ptr->link;
    while(cur!=ptr) // To print from HEAD node till END node
    {

        printf("%d*x^%d*y^%d*z^%d", cur->coef, cur->x, cur->y, cur->z);
        cur=cur->link; // move to next node
        if(cur!=ptr)
            printf("+");
    }
} // end of print

```

Void evaluate(NODE ptr) // function to evaluate the final polynomial

```

{
    Int res=0;
    Int x,y,z,ex,ey,ez,cof;
    NODE cur;
    printf("\nEnter the values of x,y,z"); // read values of X, Y and Z
    scanf("%d",&x);
    scanf("%d",&y);
    scanf("%d",&z);
    cur=ptr->link; // start with HEAD
    while(cur!=ptr) // Repeat until the end of list
    {
        ex=cur->x; // exponent of x
        ey=cur->y; // exponent of y
        ez=cur->z; // exponent of z
        cof=cur->coef; // coefficient
        res+=cof*pow(x,ex)*pow(y,ey)*pow(z,ez); // compute result for each polynomial
        cur=cur->link; // move to next node
    }
    printf("\nresult: %d", res);
} // end of evaluate

```

```
Void main(void)
{
    Int i,ch;
    NODE a=NULL,b,c;
    while(1)
    {
        printf("\n1:Represent first polynomial A");
        printf("\n2:Represent Second polynomial B");
        printf("\n3:Display the polynomial A");
        printf("\n4:Display the polynomial B");
        printf("\n5:Add A&B polynomials");//C=A+B
        printf("\n6:Evaluate polynomial C");
        printf("\n7:Exit");
        printf("\nEnter your choice:");
        scanf("%d",&ch);
        switch(ch)
        {
            case1:printf("\nEnter the elements of the polynomialA");
            a=readpoly();
            break;
            case2:printf("\nEnter the element sof th epolynomial B");
            b=readpoly();
            break;
            case 3: print(a);//display polynomial A
            break;
            case4:
                print(b);//display polynomial A
                break;
            case5:
            c=addpoly(a,b);//C=A+B
                printf("\nThe sum of two polynomialsis:");
                print(c);//display polynomial C
                printf("\n");
                break;
            case6:evaluate(c);//Evaluate polynomial C
            break;
            case7:
                return;
            default:printf("\nInvalid choice!\n");
            }//end of switch
        }//end of while
    }//end of main
```

Solution 2

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
struct node
{
int coeff;
int expo;
struct node *ptr;
};
struct node *head1,*head2,*head3, *temp,*temp1,*temp2,*temp3,*list1,*list2,*list3;
struct node *dummy1,*dummy2;
void create_poly1(int , int);
void create_poly2(int , int);
void display();
void add_poly();
void eval_poly(int );
int n,ch;
int c,e,i;

void main()
{
int x; list1=list2=NULL;
printf("1.Create first polynomial\n2.Create Second Polynomial\n3.Display both the
polynomials\n");
printf("4.Add Polynomials\n5.Evaluate a Polynomial\n6.Exit\n");
while(1)
{
printf("Enter choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1: printf("Enter the number of terms\n");
scanf("%d",&n);
printf("Enter coefficient & power of each term\n");
for(i=0;i<n;i++)
{
scanf("%d%d",&c,&e);
create_poly1(c,e);
}
break;
case 2: printf("Enter the number of terms\n");
scanf("%d",&n);
printf("Enter coefficient & power of each term\n");
for(i=0;i<n;i++)
{
scanf("%d%d",&c,&e);
create_poly2(c,e);
}
break;
```

```
        case3:Display();break;

    case4:add_poly();
        break;

        case 5:
            printf("entere value for x\n");
            scanf("%d",&x);
            eval_poly(x);
            break;
        case6: exit(0);
    }
}
}

void create_poly1(int c, int e)
{
    Dummy    1=(struct    node*)malloc(1*sizeof(struct
    node));
    dummy1->coeff=0;
    dummy1->expo=0;
    dummy1->ptr=list1;
    if(list1==NULL)
    {
        list1->coeff=c;
        list1->expo=e;
        list1->ptr=list1;
        head1=list1;
        head1->ptr=dummy1;
    }
    else
    {
        temp=(struct    node*)malloc(1*sizeof(struct
        node));
        temp->coeff=c;
        temp->expo=e;
        head1->ptr=temp;
        temp->ptr=dummy1;
    }
}
```



```
void create_poly2(int c, int e)
{
    dummy2=(struct node*)malloc(1*sizeof(struct node));
    dummy2->coeff=0;
    dummy2->expo=0;
    dummy2->ptr=list2;
    if(list2==NULL)
    {
        list2=(struct node*)malloc(1*sizeof(struct node));
        list2->coeff=c;
        list2->expo=e;
        list2->ptr=list2;
        head2=list2;
        head2->ptr=dummy2;
    }
    else
    {
        temp=(struct node*)malloc(1*sizeof(struct node));
        temp->coeff=c;
        temp->expo=e;
        head2->ptr=temp;
        temp->ptr=dummy2;
        head2=temp;
    }
}
```

```
void add_poly()
{
    temp1=list1;
    temp2=list2;
    while((temp1!=dummy1)&&(temp2!=dummy2))
    {
        temp=(struct node*)malloc(1*sizeof(struct node));
        if(list3==NULL)
        {
            list3=temp;
            head3=list3;
        }
        if(temp1->expo==temp2->expo)
        {
            temp->coeff=temp1->coeff+temp2->coeff;
            temp->expo=temp1->expo;
            temp->ptr=list3;
            head3->ptr=temp;
            head3=temp;
            temp1=temp1->ptr;
            temp2=temp2->ptr;
        }
        else if(temp1->expo>temp2->expo)
        {
            temp->coeff=temp1->coeff;
            temp->expo=temp1->expo;
            temp->ptr=list3;
```

```

        head3->ptr=temp;
        head3=temp;
        temp1=temp1->ptr;
    }
    else
    {
        temp->coeff=temp2->coeff;
        temp->expo=temp2->expo;
        temp->ptr=list3;
        head3->ptr=temp;
        head3=temp;
        temp2=temp2->ptr;
    }
}

if(temp1==dummy1)
{
    while(temp2!=dummy2)
    {
        temp=(struct node*)malloc(1*sizeof(struct node));
        temp->coeff=temp2->coeff;
        temp->expo=temp2->expo;
        temp->ptr=list3;
        head3->ptr=temp;
        head3=temp;
        temp2=temp2->ptr;
    }
}
if(temp2==dummy2)
{
    while(temp1!=dummy1)
    {
        temp=(struct node*)malloc(1*sizeof(struct node));
        emp->coeff=temp1->coeff;
        temp->expo=temp1->expo;
        temp->ptr=list3;
        head3->ptr=temp;
        head3=temp;
        temp1=temp1->ptr;
    }
}
}
}

void display()
{
    temp1=list1; temp2=list2; temp3=list3;
    printf("\nPOLYNOMIAL 1:");
    while(temp1!=dummy1)
    {
        printf("%dX^%d+",temp1->coeff,temp1->expo); temp1=temp1->ptr;
    }
    printf("\b ");
    printf("\nPOLYNOMIAL 2:");

    while(temp2!=dummy2)

```

```

    {
        printf("%dX^%d+",temp2->coeff,temp2->expo); temp2=temp2->ptr;
    }
    printf("\b ");
    printf("\n\nSUM OF POLYNOMIALS:\n");
    while(temp3->ptr!=list3)
    {
        printf("%dX^%d+",temp3->coeff,temp3->expo); temp3=temp3->ptr;
    }
    printf("%dX^%d\n",temp3->coeff,temp3->expo);
}

void eval_poly(int x)
{
    int result=0; temp1=list1; temp2=list2;
    while(temp1!=dummy1)
    {
        result+=(temp1->coeff)*pow(x,temp1->expo); temp1=temp1->ptr;
    }
    printf("Polynomial 1 Evaluation:%d\n",result); result=0;
    while(temp2!=dummy2)
    {
        result+=(temp2->coeff)*pow(x,temp2->expo); temp2=temp2->ptr;
    }
    printf("Polynomial 2 Evaluation:%d\n",result);
}

```

OUTPUT:

```

Enter your choice: 5

The sum of two polynomials is: 2*x^5*y^7*z^9 + 3*x^6*y^7*z^9 + 7*x^5*y^2*z^1 + 9
*x^5*y^3*z^1

1: Represent first polynomial A
2: Represent Second polynomial B
3: Display the polynomial A
4: Display the polynomial B
5: Add A & B polynomials
6: Evaluate polynomial C
7: Exit
Enter your choice: 6

Enter the values of x, y,z 1 1 1

result: 21
1: Represent first polynomial A
2: Represent Second polynomial B
3: Display the polynomial A
4: Display the polynomial B
5: Add A & B polynomials
6: Evaluate polynomial C
7: Exit
Enter your choice: 7

```

SOLUTION-3

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<math.h> typedef
struct node
{
int expo,coef; struct
node *next; }node;
node * insert(node *,int, int); node *
create();
node * add(node *p1,node *p2); int
eval(node *p1);
void display(node *head);
node *insert(node*head,int expo1,int coef1)
{
node *p,*q;
p=(node *)malloc(sizeof(node));
p->expo=expo1;
p->coef=coef1;
p->next=NULL;
if(head==NULL)
{
head=p;
head->next=head;
return(head);
}
if(expo1>head->expo)
{
p->next=head->next; head-
>next=p; head=p;
return(head);
}
if(expo1==head->expo)
{
head->coef=head->coef+coef1;
return(head);
}
q=head;
while(q->next!=head&&expo1>=q->next->expo) q=q-
>next;
if(p->expo==q->expo)
q->coef=q->coef+coef1; else
{
p->next=q->next; q-
>next=p;
}
```

```
return(head);
}
node *create()
{
int n,i,expo1,coef1; node
*head=NULL;
printf("\n\nEnter no of terms of polynomial==>");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\n\nEnter coef & expo==>");
scanf("%d%d",&coef1,&expo1);
head=insert(head,expo1,coef1);
}
return(head);
}
node *add(node *p1,node *p2)
{
node *p;
node *head=NULL;
printf("\n\nAddition of polynomial==>");
p=p1->next;
do
{
head=insert(head,p->expo,p->coef); p=p-
>next;
}
while(p!=p1->next); p=p2-
>next;
do
{
head=insert(head,p->expo,p->coef); p=p-
>next;
}while(p!=p2->next);
return(head);
}
int eval(node *head)
{
node *p; int
x,ans=0;
printf("\n\nEnter the value of x=");
scanf("%d",&x);
p=head->next;
do
{
ans=ans+p->coef*pow(x,p->expo); p=p-
>next;
```

```

} while(p!=head->next);
return(ans);
}
void display(node *head)
{
node *p,*q;
int n=0; q=head-
>next; p=head->next;
do
{
n++; q=q->next;
} while(q!=head->next);

printf("\n\n\tThe polynomial is==>");
Do
{
if(n-1)
{
printf("%dx^(%d) + ",p->coef,p->expo);
p=p->next;
}
Else
{
printf(" %dx^(%d)",p->coef,p->expo);
p=p->next;
}
n--;
} while(p!=head->next);
}
void main()
{
int a,x,ch;
node *p1,*p2,*p3;
p1=p2=p3=NULL;
while(1)
{
printf("\n\t-----<< MENU >>----- ");
printf("\n\tPolynomial Operations :");
printf(" 1.Add");
printf("\n\t\t\t2.Evaluate");
printf("\n\t\t\t3.Exit");
printf("\n\t.....");
printf("\n\n\tEnter your choice==>");
scanf("%d",&ch);
switch(ch)
{
case 1 :

```

```

p1=create();

display(p1);
p2=create();
display(p2);
p3=add(p1,p2);
display(p3);
break; case 2 :
p1=create();
display(p1);
a=eval(p1);
printf("\n\nValue of polynomial=%d",a); break;
case 3 :
exit(0); break;
default :
printf("\n\n\tinvalid choice"); break;
}
}
}

```

Output:

```

.....<< MENU >>.....
Polynomial Operations : 1.Add
                      2.Evaluate 3.Exit
-----

Enter your choice==>1

Enter no of terms of polynomial==>3

Enter coef & expo==>1 2
Enter coef & expo==>3 4

Enter coef & expo==>5 6

The polynomial is==>1x^(2) + 3x^(4) + 5x^(6)

Enter no of terms of polynomial==>3

Enter coef & expo==>6 5

Enter coef & expo==>4 3

Enter coef & expo==>2 1

```

The polynomial is==> $2x^{(1)} + 4x^{(3)} + 6x^{(5)}$

Addition of polynomial==>

The polynomial is==> $2x^{(1)} + 1x^{(2)} + 4x^{(3)} + 3x^{(4)} + 6x^{(5)} + 5x^{(6)}$

.....<< MENU >>.....

Polynomial Operations : 1.Add

2.Evaluate 3.Exit

Enter your choice==>2

Enter no of terms of polynomial==>3

Enter coef & expo==>1 4

Enter coef & expo==>2 5

Enter coef & expo==>3 6

The polynomial is==> $1x^{(4)} + 2x^{(5)} + 3x^{(6)}$

Enter the value of x=3

Value of polynomial=2754

-----<< MENU >>-----

Polynomial Operations : 1.Add

2.Evaluate 3.Exit

Enter your choice==>3

Process returned 0 (0x0) execution time : 50.297 s

Press any key to continue.

/*10Q.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**
- e.Exit.*/**

Theory:

A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties–

- The left sub-tree of a node has a key less than or equal to its parent node's key.
- The right sub-tree of a node has a key greater than or equal to its parent node's key.

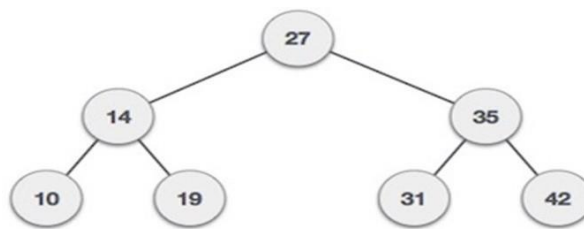
Thus, BST divides all its sub-trees into two segments; the left sub-tree and the right sub-tree and can be defined as–

$\text{left_subtree}(\text{keys}) \leq \text{node}(\text{key}) \leq \text{right_subtree}(\text{keys})$

Representation

BST is a collection of nodes arranged in a way where they maintain BST properties. Each node has a key and an associated value. While searching, the desired key is compared to the keys in BST and if found, the associated value is retrieved.

Following is a pictorial representation of BST–



We observe that the root node key(27) has all less-valued keys on the left sub-tree and the higher valued keys on the right sub-tree.

Basic Operations

Following are the basic operations of a tree–

- **Search** – Searches an element in a tree.
- **Insert** – Inserts an element in a tree.
- **Pre-order Traversal** – Traverses a tree in a pre-order manner.
- **In-order Traversal** – Traverses a tree in an in-order manner.
- **Post-order Traversal** – Traverses a tree in a post-order manner.

Traversal is a process to visit all the nodes of a tree and may print their values too. Because, all nodes are connected via edges(links) we always start from the root(head) node. That is, we cannot randomly access a node in a tree. There are three ways which we use to traverse a tree–

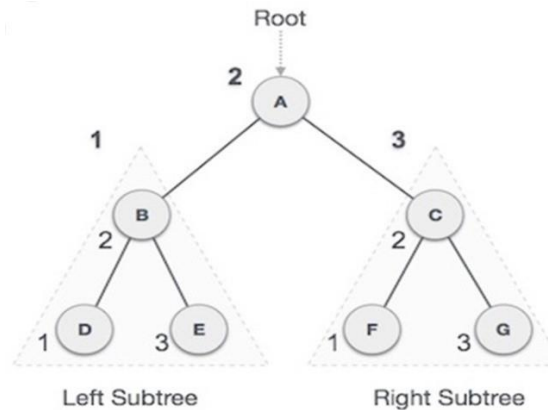
- In-order Traversal
- Pre-order Traversal
- Post-order Traversal

Generally, we traverse a tree to search or locate a given item or key in the tree or to print all the values it contains.

In-order Traversal

In this traversal method, the left sub tree is visited first, then the root and later the right sub-tree. We should always remember that every node may represent a subtree itself.

If a binary tree is traversed **in-order**, the output will produce sorted key values in an ascending



order.

We start from **A**, and following in-order traversal, we move to its left subtree **B**. **B** is also traversed in-order. The process goes on until all the nodes are visited. The output of in-order traversal of this tree will be—

$D \rightarrow B \rightarrow E \rightarrow A \rightarrow F \rightarrow C \rightarrow G$

Pre-order Traversal

In this traversal method, the root node is visited first, then the left subtree and finally the right subtree. We start from **A**, and following pre-order traversal, we first visit **A** itself and then move to its left subtree **B**. **B** is also traversed pre-order. The process goes on until all the nodes are visited. The output of pre-order traversal of this tree will be—

$A \rightarrow B \rightarrow D \rightarrow E \rightarrow C \rightarrow F \rightarrow G$

Post-order Traversal

In this traversal method, the root node is visited last, hence the name. First we traverse the left subtree, then the right subtree and finally the root node. We start from **A**, and following pre-order traversal, we first visit the left subtree **B**. **B** is also traversed post-order. The process goes on until all the nodes are visited. The output of post-order traversal of this tree will be—

$D \rightarrow E \rightarrow B \rightarrow F \rightarrow G \rightarrow C \rightarrow A$

Algorithm:

Algorithm: Binary_Search_Tree_operations

1. Start
2. Read choice for BST operations
3. Switch ch
 - case 1: Read n elements one by one and insert the m into tree
insert(item, root)
 - case 2: Traversal(root)
 - case 3: Search(root)
 - case 4: Exit
4. Stop

Algorithm:insert(item,root)

- 1.Start
- 2.Create node
temp=getnode()
temp->info=item
3. IfrootisNULL
Set temp as root
Other wise compare
temp->info with root
if t is lesser move left
otherwise
remove towards right
insert_node
4. Repeat step2 until you find leaf node
- 5.Stop

Algorithm:Pre(root)

1. Start
2. Ifroot!=NULL
Print root pre(root->llink)
pre(root->rlink)
3. Stop

Algorithm:Post(root)

- 1.Start
2. ifroot!=NULL
post(root->llink)
post(root->rlink)
print root
3. Stop

Algorithm:In(root)

- 1.Start
2. ifroot!=NULL
in(root->llink)
print root
in(root->rlink)
3. Stop

Algorithm:Traversal(root)

- 1.Start
- 2.pre(root)
- 3.post(root)
- 4.in(root)
- 5.Stop

Algorithm:Search(root)

- 1.Read key element to be inserted
- 2.if key=root->info
Print successful search else
Ifkey>root->info
root=root->rlink
elseifkey<root->info

```
        root=root>llink
    else
        print unsuccessful search
```

Solution-1

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

int flag=0;
typedef struct BST
{
int data;
struct BST *lchild, *rchild;
} node;
void insert(node *, node *);
void inorder(node *); void
preorder(node *); void
postorder(node *);
node *search(node *, int, node **);
void main()
{
int choice;
int ans=1; int key;
node *new_node, *root, *tmp, *parent;
node *get_node();
root = NULL;
printf("\nProgram For Binary Search Tree "); do {
printf("\n1.Create");
printf("\n2.Search");
printf("\n3.Recursive Traversals");
printf("\n4.Exit");
printf("\n Enter your choice :");
scanf("%d", &choice);
switch (choice)
case 1: do
{
new_node = get_node(); printf("\nEnter
The Element ");
scanf("%d", &new_node->data);
if (root == NULL) /* Tree is not Created */
root = new_node;
else
insert(root, new_node);
printf("\n Want To enter More Elements?(1/0)");
scanf("%d",&ans);
```

```
}
while (ans);
break;

case 2:
printf("\n Enter Element to be searched :");
scanf("%d", &key);
tmp = search(root, key, &parent);
if(flag==1)
{
printf("\nParent of node %d is %d", tmp->data, parent->data);
}
else
{
printf("\n The %d Element is not Present",key);
}
flag=0;
break;
case 3:
if (root == NULL)
printf("Tree Is Not Created");
else {
printf("\nThe Inorder display : ");
inorder(root);
printf("\nThe Preorder display : ");
preorder(root);
printf("\nThe Postorder display : ");
postorder(root);
}
break;
} while (choice != 4);
}
/* Get new Node */
node *get_node()
{
node *temp;
temp = (node *) malloc(sizeof(node));
temp->lchild = NULL;
temp->rchild = NULL; return
temp;
}
/*
This function is for creating a binary search tree */
void insert(node *root, node *new_node)
{
if (new_node->data < root->data)
```

```
{
if(root->lchild == NULL)
root->lchild = new_node;
else
insert(root->lchild, new_node);
}
if (new_node->data > root->data)
{
if (root->rchild == NULL)
root->rchild = new_node; else
insert(root->rchild, new_node);
}
}
/*
```

This function is for searching the node from binary Search Tree */

```
node *search(node *root, int key, node **parent)
{
node *temp;
temp = root;
while (temp != NULL)
{ if (temp->data == key)

{
printf("\n The %d Element is Present", temp->data); flag=1;
return temp;
}
*parent = temp;
if(temp->data > key)
temp = temp->lchild;
else
temp = temp->rchild;
}
return NULL;
}
/*
```

This function displays the tree in inorder fashion */

```
void inorder(node *temp)
{
if (temp != NULL)
{
inorder(temp->lchild);
printf("%d\t", temp->data);
inorder(temp->rchild);
}
}
/*
```

This function displays the tree in preorder fashion */

```
void preorder(node *temp)
{
if (temp != NULL)
{
printf("%d\t", temp->data);
preorder(temp->lchild);
preorder(temp->rchild);
}
}
/*
This function displays the tree in postorder fashion
*/
void postorder(node *temp)
{
if (temp != NULL)
{
postorder(temp->lchild);
postorder(temp->rchild);
printf("%d\t", temp->data);
}
}
```

Output:

Program For Binary Search Tree

1.Create

2.Search

3.Recursive Traversals 4.Exit

Enter your choice :1

Enter The Element 10

Want To enter More Elements?(1/0)1

Enter The Element 20

Want To enter More Elements?(1/0)1

Enter The Element 30

Want To enter More Elements?(1/0)1

Enter The Element 40

Want To enter More Elements?(1/0)1

Enter The Element 50

Want To enter More Elements?(1/0)1
Enter The Element 60

Want To enter More Elements?(1/0)1

Enter The Element 70

Want To enter More Elements?(1/0)0

1.Create
2.Search
3.Recursive Traversals
4.Exit
Enter your choice :2

Enter Element to be searched :50

The 50 Element is Present
Parent of node 50 is 40 1.Create
2.Search
3.Recursive Traversals
4.Exit
Enter your choice :2

Enter Element to be searched :55

The 55 Element is not Present
1.Create
2.Search
3.Recursive Traversals
4.Exit
Enter your choice :3

The Inorder display : 10 20 30 40 50 60 70

The Preorder display : 10 20 30 40 50 60 70

The Postorder display : 70 60 50 40 30 20 10

1.Create
2.Search
3.Recursive Traversals 4.Exit
Enter your choice :4

Process returned 4 (0x4) execution time : 88.500 s
Press any key to continue.

Solution 2:

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

struct BST
{
    int data;
    struct BST *lchild;
    struct BST *rchild;
};
typedef struct BST * NODE;

NODE get_node()
{
    NODE temp;
    temp = (NODE) malloc(sizeof(struct BST));
    temp->lchild = NULL;
    temp->rchild = NULL;
    return temp;
}

void insert(NODE root, NODE newnode);
void inorder(NODE root);
void preorder(NODE root);
void postorder(NODE root);
void search(NODE root, int key);

void insert(NODE root, NODE newnode)
{
    /*Note: if newnode->data == root->data it will be skipped. No duplicate nodes are allowed */

    if (newnode->data < root->data)
    {
        if (root->lchild == NULL)
            root->lchild = newnode;
        else
            insert(root->lchild, newnode);
    }
    if (newnode->data > root->data)
    {
        if (root->rchild == NULL)
            root->rchild = newnode;
        else
            insert(root->rchild, newnode);
    }
}

void search(NODE root, int key)
{
    NODE cur;
    if(root == NULL)
    {
        printf("\nBST is empty.");
    }
}
```

```
        return;
    }
    cur = root;
    while (cur != NULL)
    {
        if (cur->data == key)
        {
            printf("\nKey element %d is present in BST", cur->data); return;
        }
        if (key < cur->data)
            cur = cur->lchild;
        else
            cur = cur->rchild;
    }
    printf("\nKey element %d is not found in the BST", key); }
```

```
void inorder(NODE root)
{
    if(root != NULL)
    {
        inorder(root->lchild);
        printf("%d ", root->data);
        inorder(root->rchild);
    }
}
```

```
void preorder(NODE root)
{
    if (root != NULL)
    {
        printf("%d ", root->data);
        preorder(root->lchild);
        preorder(root->rchild);
    }
}
```

```
void postorder(NODE root)
{
    if (root != NULL)
    {
        postorder(root->lchild);
        postorder(root->rchild);
        printf("%d ", root->data);
    }
}
```

```
void main()
{
    int ch, key, val, i, n;
    NODE root = NULL, newnode;
    while(1)
    {
        printf("\n~~~~~BST MENU~~~~~");
```

```

printf("\n1.Create a BST");
printf("\n2.Search");
printf("\n3.BST Traversals: ");
printf("\n4.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)
{
    case 1:    printf("\nEnter the number of elements: ");
               scanf("%d", &n);
               for(i=1;i<=n;i++)
               {
                   newnode = get_node();
                   printf("\nEnter The value: ");
                   scanf("%d", &val);
                   newnode->data = val;
                   if (root == NULL)
                       root = newnode;
                   else
                       insert(root, newnode);
               }
               break;
    case 2:    if (root == NULL)
               printf("\nTree Is Not Created");
               else
               {
                   printf("\nThe Preorder display : ");
                   preorder(root);
                   printf("\nThe Inorder display : ");
                   inorder(root);
                   printf("\nThe Postorder display : ");
                   postorder(root);
               }
               break;
    case 3:    printf("\nEnter Element to be searched: ");
               scanf("%d", &key);
               search(root, key);
               break;

    case 4:    exit(0);
}
}
}

```

ALGORITHM:

- Step 1: Start.
- Step 2: Create a Binary Search Tree for N elements.
- Step 3: Traverse the tree in inorder.
- Step 4: Traverse the tree in preorder
- Step 6: Traverse the tree inpostorder.

Step 7: Search the given key element in theBST.

Step 8: Delete an element fromBST.

Step 9: Stop

Solution 3:

```
#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;
}

NODE* del(NODE *node, int data)
{
    NODE *temp;
    if(node == NULL)
    {
        printf("\nElement not found");
    }
    else if(data < node->data)
    {
        node->left = del(node->left, data);
    }
    else if(data > node->data)
    {
        node->right = del(node->right, data);
    }
}
```

```

        else
        {
/* Now We can delete this node and replace with either minimum element in the right sub tree or
maximum element in the left subtree */
        if(node->right && node->left)
        {
/* Here we will replace with minimum element in the right sub tree */
            temp = findMin(node->right);
            node -> data = temp->data;
/* As we replaced it with some other node, we have to delete that node */
            node -> right = del(node->right, temp->data);
        }
    else
    {
/* If there is only one or zero children then we can directly remove it from the tree and connect its
parent to its child */
        temp = node;
        if(node->left == NULL)
            node=node->right;
        else
            if(node->right ==NULL)
                node = node->left;
        free(temp);    /* temp is longer required */
    }
}
return node;
}
void main()
{
    int data, ch, i,
    n; NODE
    *root=NULL;
    clrscr();
    while (1)
    {
        printf("\n1.Insertion in Binary Search Tree")
        printf("\n2.Search Element in Binary Search Tree");
        printf("\n3.Delete 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);
        break;
    case 3:printf("\nEnter the element to delete: ");
            scanf("%d", &data);
            root=del(root, data);
            break;
case4:

    printf("\nInorder Traversal: \n");
    inorder(root);
    break;

case5:

    printf("\nPreorder Traversal: \n");
    preorder(root);
    break;
```

case 6:

```
printf("\nPostorder Traversal: \n");
postorder(root);
break;
```

case 7: k;

```
}
}
}
```

```
exit(0);
```

default

```
printf ("\nWrongoption");
break;
```

OUTPUT

```
1. Create BST of N Integers
2. BST Traversal
3. Binary Search
4. Exit
Enter Your Choice: 1

Enter number elements : 12
Enter the item to be inserted
6
Enter the item to be inserted
9
Enter the item to be inserted
5
Enter the item to be inserted
2
Enter the item to be inserted
8
Enter the item to be inserted
15
Enter the item to be inserted
24
Enter the item to be inserted
14
Enter the item to be inserted
7
Enter the item to be inserted
8
Enter the item to be inserted
5
Enter the item to be inserted
2
```



```
1. Create BST of N Integers
2. BST Traversal
3. Binary Search
4. Exit
Enter Your Choice: 2

PREORDER traversal
6      5      2      9      8      7      15      14      24

INORDER traversal
2      5      6      7      8      9      14      15      24

POSTORDER traversal
2      5      7      8      14      24      15      9      6

1. Create BST of N Integers
2. BST Traversal
3. Binary Search
4. Exit
Enter Your Choice: 3
enter the element to be searched
24
Found key 24 in tree

1. Create BST of N Integers
2. BST Traversal
3. Binary Search
4. Exit
Enter Your Choice: 3
enter the element to be searched
30
Key not found
```

```
1. Create BST of N Integers
2. BST Traversal
3. Binary Search
4. Exit
Enter Your Choice: 4
```

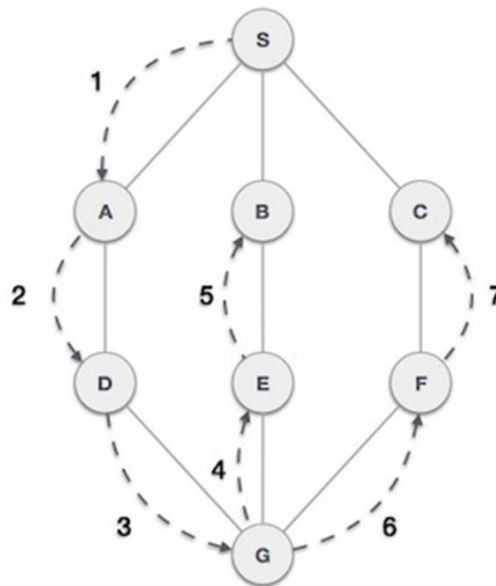
PROGRAM-11

/*11Q.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 DFS/BFS method.*/

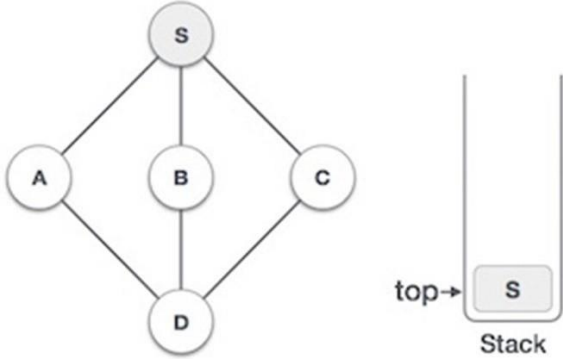
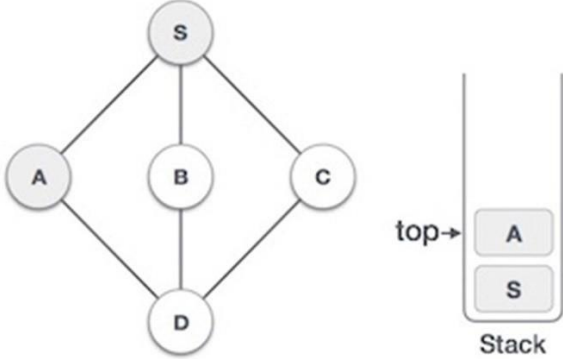
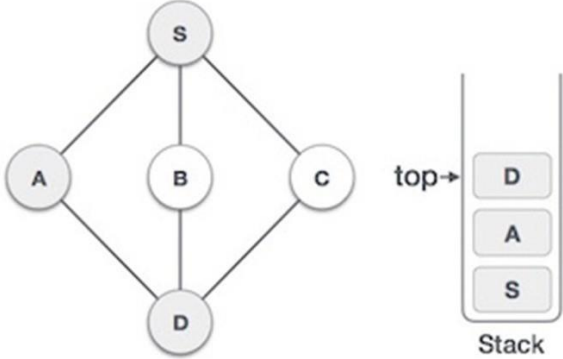
Depth First Search(DFS) algorithm traverses a graph in a depth ward motion and uses a stack to remember to get the next vertex to start a search,when a dead end occurs in any iteration.

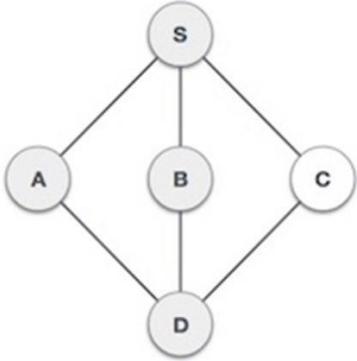
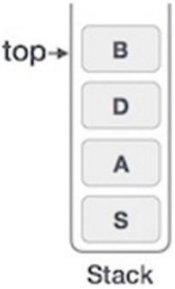
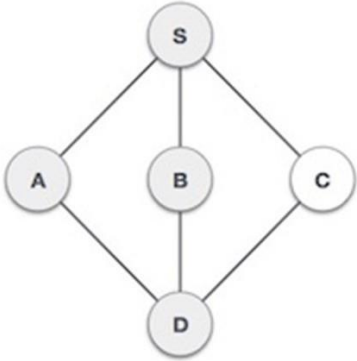

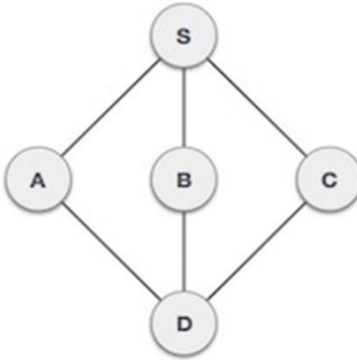



As in the example given above,DFS algorithm traverses from A to B to C to D first then to E,then to F and lastly to G. It employs the following rules.

- **Rule1**–Visit the adjacent unvisited vertex.Mark it as visited.Display it.Push it In a stack.
- **Rule2**–If no adjacent vertex is found,pop up a vertex from the stack.(It will pop up all the Vertices from the stack,which do not have adjacent vertices.)
- **Rule3**–Repeat Rule1 and Rule2 until the stack is empty.

Step	Traversal	Description
1.		Initialize the stack.

2.		<p>Mark S as visited and put it onto the stack. Explore any unvisited adjacent node from S. We have three nodes and we can pick any of them. For this example, we shall take the node in alphabetical order.</p>
3.		<p>Mark A as visited and put it onto the stack. Explore any unvisited adjacent node from A. Both S and D are adjacent To A but we are concerned for unvisited nodes only.</p>
4.		<p>Visit D and mark it as visited and put onto the stack. Here, we have B and C nodes, which are adjacent to D and both are unvisited. However, we shall again choose in alphabetical order.</p>

5.	 	<p>We choose B, mark it as visited and put onto the stack.</p> <p>Here B does not have any unvisited adjacent node. So, we pop B from the stack.</p>
6.	 	<p>We check the stack top for return to the previous node and check if it has any unvisited nodes. Here, we find D to be on the top of the stack.</p>
7.	 	<p>Only unvisited adjacent node is from D is C now. So we visit C, mark it as visited and put it onto the stack.</p>

Algorithm:***Algorithm Graph-Of-Cities***

1. Start
2. Read no of cities
3. Read adjacent matrix for cities
4. Read choice for graph operations
5. Switch choice

Case1: Read source vertex

Reachability(v)
Print reachable nodes from source vertex

Case2:Connectivity(1)
If all vertex are visited then print graph is connected
else Graph is disconnected

Case3:Exit

Algorithm:Reachability(v)

1. Start
2. Repeat until n
 - if(a[v][i] and! visited[i])
 - pushs[++top]=i;
 - if(top>=0)
 - then visited(s[top])=1
- reachability(s[top--])
3. Stop

Algorithm:Connectivity(r)

1. Start
2. Repeat until n.
 - if(a[v][i] and !reach[i])print v to I
 - connectivity connectivity(i)
3. Stop.

SOLUTION-1

```
#include<stdio.h>
Inta[20][20],q[20],visited[20],reach[10];intn,i,j,count,f=0,r=-1;

Void bfs(int v)//Reachability using BreadthFirstSearch
{
    for(i=1;i<=n;i++)
        if(a[v][i]&&!reach[i])//check weather node is reached
            q[++r]=i;//if not add it TO Queue
            if(f<=r)//check for non empty Queue
            {
```

```

        q[f]=1;//
        bfs(q[f++]); //recursive call B
    }

    printf("\nPostorder Traversal: \n");
    postorder(root);
    break;
}
}
}

    exit(0);

} //end of BSF

void dfs(int v) //Connectivity using DepthFirstSearch
{
    int i;
    visited[v]=1; //set source is visited
    for(i=1; i<=n; i++)
    {
        if(a[v][i] && !visited[i]) //check whether node is visited
        {
            printf("\n%d>%d", v, i); //if not print count++ visited
            count dfs(i); //recursive call DFS
        }
    }
} //end of DFS

Void main()
{
    Int v, choice;
    printf("\nEnter the number of cities:");
    scanf("%d", &n);
    printf("\nEnter graph data in matrix form:\n");
    for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
            scanf("%d", &a[i][j]); //read adjacency matrix
    while(1)
    {
        printf("\n1. Test for reachability");
    }
}

```

```

printf("\n2.      Test      for      connectivity")
;printf("\n3.Exit");
printf("\nEnter Your Choice:");
scanf("%d",&choice);
switch(choice)
{
    case1:printf("Enter the source vertex:");
           scanf("%d",&v);
if((v<1)||(v>n))//check for valid source entry
           printf("\nEnter a valid source vertex");

else//if valid begin test for reachability
    {

        for(i=1;i<=n;i++)//begin with assuming all cities

        reach[i]=0;
        reach[v]=1;//source is reached
        bfs(v);//cal BFS to check reachability
        printf("The reachable nodes from node%d:\n",v);
        for(i=1;i<=n;i++)//display reachable cities from

        if(reach[i]&& i!=v)
            printf("node%d\n",i);
        }
        break;

    case2:
        for(i=1;i<=n;i++)//begin with assuming all cities are not connected
        visited[i]=0;
        printf("ENTER the source vertex");
        count=0;
        scanf("%d",&v);//read source city
        dfs(v);//calDFS to check connectivity
        if(count==n-1)
            printf("\nGraph is connected\n");
        else
            printf("\nGraph is notconnected");
        break;

    case3:return;
    default:printf("\nEnter proper Choice");
} //end of switch
} //end of while
} //end of main

```

Solution 2:**ALGORITHM:**

Step 1: Start.

Step 2: Input the value of N nodes of the graph

Step 3: Create a graph of N nodes using adjacency matrix representation.

Step 4: Print the nodes reachable from the starting node using BFS.

Step 5: Check whether graph is connected or not using DFS.

Step 6: Stop.

PROGRAM CODE:

```
#include <stdio.h>
#include <stdlib.h>
int a[20][20],q[20],visited[20],reach[10],n,i,j,f=0,r=-1,count=0;
void bfs(int v)
{
    for(i=1;i<=n;i++)
        if(a[v][i] && !visited[i]) q[++r]=i;
    if(f<=r)
    {
        visited[q[f]]=1;
        bfs(q[f++]);
    }
}
void dfs(int v)
{
    int i; reach[v]=1;
    for(i=1;i<=n;i++)
    {
        if(a[v][i] && !reach[i])
        {
            printf("\n %d->%d",v,i);
            count++;
            dfs(i);
        }
    }
}

void main()
{
```



```
int v, choice;
printf("\n Enter the number of vertices:");
scanf("%d",&n);

for(i=1;i<=n;i++)
{
    q[i]=0;
    visited[i]=0;
}
for(i=1;i<=n-1;i++)
reach[i]=0;
printf("\n Enter graph data in matrix form:\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
    scanf("%d",&a[i][j]);
printf("1.BFS\n 2.DFS\n 3.Exit\n");
scanf("%d",&choice);
switch(choice)
{
    case 1:
        printf("\n Enter the starting vertex:");
        scanf("%d",&v);
        bfs(v);
        if((v<1)|| (v>n))
        {
            printf("\n Bfs is not possible");
        }
        else
        {
            printf("\n The nodes which are reachable from %d:\n",v);
            for(i=1;i<=n;i++)
                if(visited[i]) printf("%d\t",i);
        }
        break;

    case 2:
        dfs(1);
        if(count==n-1)
            printf("\n Graph is connected");
        else
            printf("\n Graph is not connected");
            break;
```

```
        case 3: exit(0);
    }
}
```

Solution 3:

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

```
void bfs(int v);
void dfs(int v);
```

```
int a[50][50], n, visited[50];
int q[20], front = -1, rear = -1;
int s[20], top = -1, count=0;
```

```
void creategraph()
{
    int i, j;
    printf("\nEnter the number of vertices in graph: ");
    scanf("%d", &n);
    printf("\nEnter the adjacency matrix:\n");
    for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
            scanf("%d", &a[i][j]);
}
```

```
void bfs(int v)
{
    int i, cur;
    visited[v] = 1;
    q[++rear] = v;
    printf("\nNodes reachable from starting vertex %d are: ", v);
    while(front!=rear)
    {
        cur = q[++front];
        for(i=1; i<=n; i++)
        {
            if((a[cur][i]==1)&&(visited[i]==0))
            {
                q[++rear]=i;
                visited[i]=1;
                printf("%d ", i);
            }
        }
    }
}
```

```

}

void dfs(int v)
{
    int i;
    visited[v]=1;
    s[++top] = v;
    for(i=1;i<=n;i++)
    {
        if(a[v][i] == 1&& visited[i] == 0 )
        {
            dfs(i);
            count++;
        }
    }
}

int main()
{
    int ch, start, i, j;
    creategraph();
    printf("\n\n~~~Menu~~~");
    printf("\n==>1. BFS: Print all nodes reachable from a given starting node");
    printf("\n==>2. DFS: Print all nodes reachable from a given starting node");
    printf("\n==>3:Exit");
    printf("\nEnter your choice: ");
    scanf("%d", &ch);
    switch(ch)
    {
        case 1:
            for(i=1;i<=n;i++)
                visited[i] = 0;
            printf("\nEnter the starting vertex: ");
            scanf("%d", &start);
            bfs(start);
            for(i=1;i<=n;i++)
            {
                if(visited[i] == 0)
                    printf("\nThe vertex that is not reachable is %d", i);
            }
            break;
        case 2:
            for(i=1;i<=n;i++)
                visited[i] = 0;
            printf("\nEnter the starting vertex:\t");
            scanf("%d", &start);

```

```

        dfs(start);
        printf("\nNodes reachable from starting vertex %d are:\n", start);
        for(i=1; i<=count; i++)
            printf("%d\t", s[i]);
        break;
    case 3: exit(0);
    default: printf("\nPlease enter valid choice:");
}
}

```

OUTPUT:

```

Enter the number of cities: 5
Enter graph data in matrix form:
0 1 1 1 0
0 0 0 0 1
0 1 0 0 0
0 1 1 0 0
1 0 0 1 0

-----Menu-----
1.Test for reachability
2.Test for connectivity
3.Exit
Enter Your Choice: 1
Enter the source vertex: 1
The reachable nodes from node 1:
node 1
node 2
node 3
node 4
node 5

-----Menu-----
1.Test for reachability
2.Test for connectivity
3.Exit
Enter Your Choice: 2
1->2
2->5
5->4
4->3
Graph is connected

-----Menu-----
1.Test for reachability
2.Test for connectivity
3.Exit
Enter Your Choice: 3

```

PROGRAM-12

12. Given a File of N employee records with a set K of Keys(4-digit) which uniquely determinethe records in file F. Assume that file F is maintained in memory by a Hash Table(HT) of mmemory locations with Lasthesetofmemoryaddresses (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 functionH: $K \rightarrow L$ as $H(K) = K \bmod m$ (remainder method), and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any)using linear probing

Theory:

Hash Table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes very fast if we know the index of the desired data.

Thus, it becomes a data structure in which insertion and search operations are very fast irrespective ofthe size of the data. Hash Table uses an array as a storage medium and uses hash technique to generate an index where an element is to be inserted or is to be located from.

Hashing

Hashing is a technique to convert a range of key values into a range of indexes of an array. We're going to use modulo operator toget a range of key values. Consider an example of hash table of size 20, and the following items are to be stored.Item are in the (key,value)format.

Sr.No.	Key	Hash	ArrayIndex
1	1	$1 \% 20 = 1$	1
2	2	$2 \% 20 = 2$	2
3	42	$42 \% 20 = 2$	2
4	4	$4 \% 20 = 4$	4
5	12	$12 \% 20 = 12$	12
6	14	$14 \% 20 = 14$	14
7	17	$17 \% 20 = 17$	17
8	13	$13 \% 20 = 13$	13
9	37	$37 \% 20 = 17$	17

Linear Probing

As we can see, it may happen that the hashing technique is used to create an already used index of the array. In such a case, we can search the next empty location in the array by looking into the next cell until we find an empty cell. This technique is called linear probing.

Basic Operations

Following are the basic primary operations of a hash table.

- **Search** –Searches an element in a hash table.
- **Insert**–inserts an element in a hash table.
- **Delete**–Deletes an element from a hash table.

Sr.No.	Key	Hash	ArrayIndex	AfterLinearProbing,ArrayIndex
1	1	$1 \% 20 = 1$	1	1
2	2	$2 \% 20 = 2$	2	2
3	42	$42 \% 20 = 2$	2	3
4	4	$4 \% 20 = 4$	4	4
5	12	$12 \% 20 = 12$	12	12
6	14	$14 \% 20 = 14$	14	14
7	17	$17 \% 20 = 17$	17	17
8	13	$13 \% 20 = 13$	13	13
9	37	$37 \% 20 = 17$	17	18

Algorithm:

Algorithm:HashTable_Linear Probing

- 1.start
- 2.read choice for HashTable operations
- 3.Switch choice
 - Case1:linear_prob(a,num)
 - Case2:display(a)
 - Case3:return
- 4.stop

Algorithm:linear_prob(a,num)

```
1. start
2. repeat until ans='y'
3. 'read 4 digit number
   Find key=create(num)
   Ifa[key]=-1
       then
           a[key]=num
   otherwise
       display Ccollision C detected
       repeat until max
       ifa[i]!=-1count++
       if count=max
       then
           display(a)return
       repeat from key+1 to max
       ifa[i]=-1
       then
           a[i]=num
           flag=1
       repeat until key and flag=0
       ifa[i]=-1
       then a[i]=num
       flag=1
   read choice 'y/n'
3.stop
```

Algorithm:create(num)

```
1.start
2.key=num%103
3.stop
```

Algorithm:display(a)

```
1. Start
2. Repeat until max displaya[i]
3. stop
```


SOLUTION-1

```
#include<stdio.h>
#define MAX10

Void linear_prob(inta[MAX],intnum)
{
    Int flag,i,key,count;
    Char ans;
    do
    {
        flag=0;
        count=0;
        printf("\nEnter 4 digit Key:");
        scanf("%4d",&num);//readanumber
        key=num%10;//generate single digit key for given number if(a[key]==1)//check for
        empty entry in Hasht able
            [key]=num;//if yes
    Else //if entry exists then avoid collision
        {
            printf("\nCollision Detected...!!!\n");i=0;
            while(i<MAX)//check for next available empty location in HT
            {
                if(a[i]!=-1)
                    count++;//count empty locationi n HT
                i++;
            }//end of while
            if(count==MAX)//if HT is full then display HT and return
            {
                printf("\nHashtable is full\n");
                display(a);//Display HT
                return;
            }//endif
            printf("Collision avoided successfully using LINEAR PROBING\n");
            for(i=key+1;i<MAX;i++)//if there is empty space after key in HT then make a entry in HT
                if(a[i]==-1)
                {
                    a[i]=num;
```

```

        flag=1;
        break;
    } //end of if
    i=0;
    while((i<key)&&(flag==0)) //check for empty space before key in HT then make a
entry in HT
    {
        if(a[i]==-1)
        {
            a[i]=num;
            flag=1;
            break;
        } //end of ifi
        ++;
    } //end of while
} //end of else
printf("\nDo you wish to continue?(y/n)");
fpurge(stdin);
scanf("%c",&ans);
}
while(ans=='y'||ans=='Y');//end of do while
} //end of linearprobe

```

```

Void display(inta[MAX]) //display the HT
{
    Int i;
    printf("\nthe hashtable is\n Key\t Value");
    for(i=0; i<MAX;i++)
        printf("%d\t%d\n",i,a[i]);
} //end of display

```

```

Void main()
{
    Int a[MAX],num,i,choice;
    for(i=0;i<MAX;i++) /initialize HT with no entries a[i]=-1;
    while(1)
    {
        printf("Collision handling by Linear Probing");
        printf("\n1.Insert into Hashtable");
    }
}

```

Solution 2

164

```
        }
        ht[hashindex] = key;
        elecount++;
    }

void displayHashTable()
{
    int i;
    if(elecount == 0)
    {
        printf("\nHash Table is empty");
        return;
    }
    printf("\nHash Table contents are:\n\n ");
    for(i=0; i<m; i++)
        printf("\nT[%d] --> %d ", i, ht[i]);
}

void main()
{
    int i;
    printf("\nEnter the number of employee records (N) : ");

    scanf("%d", &n);
    printf("\nEnter the four digit values (K) of 'N' Employee Records:\n ");
    for(i=0; i<n; i++)
        scanf("%d", &key[i]);
    printf("\nEnter the two digit memory locations (m) for hash table: ");
    scanf("%d", &m);
    createHashTable();
    printf("\nInserting key values of Employee records into hash table..... ");
    for(i=0; i<n; i++)
    {
        if(elecount == m)
        {
            printf("\nHash table is full. Cannot insert the %d record key value", i+1);
            break;
        }
        insertIntoHashTable(key[i]);
    }
    displayHashTable();
}
```

Solution 3**ALGORITHM:**

Step 1: Start.

Step 2: Given a File of N employee records with a set K of Keys (4-digit) which uniquely determine the records in file F.

Step 3: 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.

Step 4: Let the keys in K and addresses in L are Integers

Step 5: Hash function H: $K \rightarrow L$ as $H(K) = K \bmod m$ (remainder method)

Step 6: Hashing as to map a given key K to the address space L, Resolve the collision (if any) is using linear probing.

Step 7: Stop.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
int create(int);
void display (int[]);
void main()
{
    int
    a[MAX], num, key, i;
    int ans=1;
    printf(" collision handling by linear probing : \n");
    for (i=0; i<MAX; i++)
    {
        a[i] = -1;
    }
    do
    {
        printf("\n Enter the data");
        scanf("%4d", &num); key=create(num);
        linear_prob(a, key, num);
        printf("\n Do you wish to continue ? (1/0) ");
        scanf("%d", &ans);
    } while(ans);
    display(a);
}
```

```
int create(int num)
{
    int key;
    key=num%100;
    return key;
}

void linear_prob(int a[MAX], int key, int num)
{
    int flag, i, count=0;
    flag=0;
    if(a[key]== -1)
    {
        a[key] = num;
    }
    else
    {
        printf("\nCollision Detected...!!!\n");
        i=0;
        while(i<MAX)
        {
            if (a[i]!=-1)
                count++; i++;
        }
        printf("Collision avoided successfully using LINEAR PROBING\n");
        if(count == MAX)
        {
            printf("\n Hash table is full");
            display(a);
            exit(1);
        }
        for(i=key+1; i<MAX; i++)
        if(a[i] == -1)
        {
            a[i] = num;
            flag =1;
            break;
        }
        //for(i=0;i<key;i++) i=0;
        while((i<key) && (flag==0))
        {
            if(a[i] == -1)
            {
                a[i] = num;
                flag=1;
            }
        }
    }
}
```

```
                break;
            }
        i++;
    }
}

void display(int a[MAX])
{
    int i,choice;
    printf("1.Display ALL\n 2.Filtered Display\n");
    scanf("%d",&choice);
    if(choice==1)
    {
        printf("\n the hash table is\n");
        for(i=0; i<MAX; i++)
            printf("\n %d %d ", i, a[i]);
    }
    else
    {
        printf("\n the hash table is\n");
        for(i=0; i<MAX; i++) if(a[i]!=-1)
        {
            printf("\n %d %d ", i, a[i]);
            continue;
        }
    }
}
```

OUTPUT:

collision handling by linear probing :

Enter the data: 1

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

Enter the data: 2

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

Enter the data: 3

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

Enter the data: 4

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

Enter the data: 2

Collision Detected...!!!

Collision avoided successfully using LINEAR PROBING

Do you wish to continue ? (1/0) 0 1.Display

ALL

2. Filtered Display

Enter Your Choice:2

the hash table is

1	1
2	2
3	3
4	4
5	2

Process returned -1 (0xFFFFFFFF) execution time : 18.141 s

Press any key to continue

Content Beyond Syllabi

1. *Write a C Program to check whether two given lists are containing the same data.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int num;
    struct node *next;
};
void feedmember(struct node **);
int compare (struct node *, struct node *);
void release(struct node **);
int main()
{
    struct node *p = NULL;
    struct node *q = NULL;
    int result;
    printf("Enter data into first list\n");
    feedmember(&p);
    printf("Enter data into second list\n");
    feedmember(&q);
    result = compare(p, q);
    if (result == 1)
        printf("The 2 list are equal.\n");
    else
        printf("The 2 lists are unequal.\n");

    release (&p);release (&q);return 0;
}
int compare (struct node *p, struct node *q)
{
    while (p != NULL && q != NULL)
    {
        if (p->num != q-> num)
        { return 0;
        }
    }
    else
    {
        p = p->next;q = q->next;
    }
}
if (p != NULL || q != NULL) {
    return 0;
}
else {
    return 1;
}
```

```

    }
}
void feedmember (struct node **head)
{
    int c, ch;
    struct node
    *temp;
    do {
        printf("Enter number: ");scanf("%d", &c);
        temp = (struct node *)malloc(sizeof(struct node));
        temp->num = c;
        temp->next = *head;
        *head = temp;
        printf("Do you wish to continue [1/0]: ");
        scanf("%d", &ch);
    }while (ch != 0);
    printf("\n");
}
void release (struct node **head)
{ struct node *temp = *head;
  while ((*head) != NULL)
  {
    (*head) = (*head)->next;free(temp);
    temp = *head;
  }
}

```

INPUT/ OUTPUT

```

Enter data into first list
Enter number: 12
Do you wish to continue [1/0]: 1
Enter number: 3
Do you wish to continue [1/0]: 1
Enter number: 28
Do you wish to continue [1/0]: 1
Enter number: 9
Do you wish to continue [1/0]: 0
Enter data into second list
Enter number: 12
Do you wish to continue [1/0]: 1
Enter number: 3
Do you wish to continue [1/0]: 1
Enter number: 28
Do you wish to continue [1/0]: 1
Enter number: 9
Do you wish to continue [1/0]: 0
The 2 list are equal.

```

2. Write a C program to count the number of nodes in the binary search tree.

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
#define new_node (struct node*)malloc(sizeof (struct node))
struct node { int data; struct node *lchild;
struct node *rchild;
};
struct node *create_bin_rec();
void print_bin_pre_rec(struct node *t);
void cnt_nodes(struct node *t, int *l, int *nl);
void main()
{
struct node *root;
int leaf, nonleaf;
clrscr();
printf("\nCreate a binary tree \n");
root = create_bin_rec();
printf("\n Binary tree is ");
print_bin_pre_rec(root);
leaf = nonleaf = 0;
cnt_nodes(root, &leaf, &nonleaf);
printf("\n Total no. of leaf nodes are : %d ", leaf);
printf("\n Total no. of nonleaf nodes are : %d ", nonleaf);
printf("\n Total no. of nodes are : %d ", (leaf+nonleaf));
} //
main struct node *create_bin_rec()
{
int data;
struct node *t;
printf("\nData ( -1 to exit ) : ");
scanf("%d", &data);
if( data == -1) return(NULL);
else
{
t = new_node;
t->data = data;
t->lchild = create_bin_rec();
```

```
t->rchild =create_bin_rec(); return(t);
}
else
{
    create void print_bin_pre_rec(struct node *t)
    {
        if( t != NULL)
        {
            printf("%4d",t->data);
            print_bin_pre_rec(t->lchild);
            print_bin_pre_rec(t->rchild);
        } //if
    }
    // print bin pre rec
void cnt_nodes(struct node *t, int *l, int *nl)
{
    if( t != NULL)
    {
        if( t->lchild == NULL && t->rchild == NULL)
        (*l)++;
    }
    else
    (*nl)++;
    cnt_nodes(t->lchild,l,nl);
    cnt_nodes(t->rchild,l,nl);
} // if } // cnt nodes
```

output

Create binary tree

Data (-1 to exit) 10

Data (-1 to exit) 20

Data (-1 to exit) -1

Binary tree is 10 20

Total no. of leaf nodes are 1

Total no. of non leaf nodes are 1

Total no. of nodes are 2

VIVA QUESTIONS:

1. What is the need of data structures in programming?
2. Which are the basic data structures?
3. Differentiate between data structures and datatypes?
4. Differentiate between linear and nonlinear data structures?
5. What do you mean by pointer?
6. How to add two polynomials?
7. What is stack datastructure?
8. What is queue datastructure?
9. What are the applications of stack?
10. What are the applications of queue?
11. What is circular queue?
12. What is linked list?
13. What are the applications of linkedlist?
14. What are trees?
15. Differentiate graph and tree?
16. Differentiate binary tree and binary searchtree?
17. Types of binarytrees.
18. Differences between Static allocation and dynamic allocation.
19. Difference between malloc() and calloc()?
20. Define DataStructures
21. Difference between Stacks and Queues.
22. What is dangling pointer.
23. Define complete binarytree
24. Define array, what are the advantages and disadvantages of array.
25. What is the drawback of Queues,
26. What is drawback of Circular Queue
27. Define tree traversal
28. What is file?
29. Difference between late binding and early binding.
30. Define recursion
31. Define structure.
32. Differentiate between Structure and Union.