MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Belawadi, S.R Patna, Mandya -571477.

DATA STRUCTURES LABORATORY [17CSL38]
3rd SEMESTER

Academic Year: 2018-2019
(ODD Semester)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Maharaja Institute of Technology Mysore

Vision

" To be recognized as a premier technical and management institution promoting extensive education fostering research, innovation and entrepreneurial attitude"

Mission

1. To empower students with indispensable knowledge through dedicated teaching and collaborative learning.
2. To advance extensive research in science, engineering and management disciplines.
3. To facilitate entrepreneurial skills through effective institute-industry collaboration and interaction with alumni.
4. To instill the need to uphold ethics in every aspect.
5. To mould holistic individuals capable of contributing to the advancement of the society.

Department of Computer Science and Engineering

Vision

"To be a leading academic department offering computer science and engineering education, fulfilling industrial and societal needs effectively."

Mission

M1 : To enrich the technical knowledge of students in diversified areas of Computer Science and Engineering by adopting outcome based approaches.

M2 : To empower students to be competent professionals maintaining ethicality.

M3 : To facilitate the development of academia-industry collaboration.

M4 : To create awareness of entrepreneurship opportunities.

Program Educational Objectives Statements

PEO1	Be successful in solving engineering problems associated with computer science and engineering domains
PEO2	Work collaboratively on multidisciplinary projects and acquire high levels of professionalism backed by ethics
PEO3	Communicate effectively and exhibit leadership qualities, team spirit necessary for a successful career in either industry, research or entrepreneurship
PEO4	Continue to learn and advance their career through participation in the activities of professional bodies, obtaining professional certification, pursue of higher education

Program Specific Outcome (PSO)

PSO 1: Apply software engineering practices and strategies in diversified areas of computer science for solving problems using open source environment.

PSO 2: Develop suitable algorithms and codes for applications in areas of cognitive technology, computer networks with software engineering principles and practices.

INDEX PAGE		
Experiment Number	**Laboratory Experiment**	**Page Number**
	Do's and Don'ts in lab	
	VTU Lab Syllabus	
	CO's with mapping to PO's and PSO's	
01	Design, Develop and Implement a menu driven Program in C for the following **Array** operations a. Creating an Array of **N** Integer Elements b. Display of Array Elements with Suitable Headings c. Inserting an Element (**ELEM**) at a given valid Position (**POS**) d. Deleting an Element at a given valid Position(**POS**) e. Exit. Support the program with functions for each of the above operations.	01
02	Design, Develop and Implement a Program in C for the following operations on **Strings** a. Read a main String (**STR),** a Pattern String (**PAT**) and a Replace String (**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.	05
03	Design, Develop and Implement a menu driven Program in C for the following operations on **STACK** of Integers (Array Implementation of Stack with maximum size **MAX**) a. ***Push*** an Element on to Stack b. ***Pop*** an Element from Stack *c.* Demonstrate how Stack can be used to check ***Palindrome*** d. Demonstrate ***Overflow*** and ***Underflow*** situations on Stack e. Display the status of Stack f. Exit Support the program with appropriate functions for each of the above operations	06
04	Design, Develop and Implement a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: **+, -, *, /, %(Remainder), ^(Power)** and **alphanumeric** operands**.**	08
05	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	09

06	Design, Develop and Implement a menu driven Program in C for the following operations on **Circular QUEUE** of Characters (Array Implementation of Queue with maximum size **MAX**) a. Insert an Element on to Circular QUEUE b. Delete an Element from Circular QUEUE c. Demonstrate ***Overflow*** and ***Underflow*** situations on Circular QUEUE d. Display the status of Circular QUEUE e. Exit Support the program with appropriate functions for each of the above operations	11
07	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*** 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	13
08	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*** 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	16
09	Design, Develop and Implement a Program in C for the following operationson **Singly Circular Linked List (SCLL)** with header nodes a. Represent and Evaluate a Polynomial **P(x,y,z) = 6x2y2z-4yz5+3x3yz+2xy5z-2xyz3** 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	20
10	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 Post Order c. Search the BST for a given element (**KEY**) and report the appropriate message e. Exit	23

General Lab Guidelines:

- Maintain laboratory etiquettes during the laboratory sessions.
- Do not wander around or distract other students or interfere with the conduction of the experiments of other students.
- Keep the laboratory clean, do not eat, drink or chew gum in the laboratory.

DO'S

- Sign the log book when you enter/leave the laboratory.
- Read the hand out/procedure before starting the experiment. If you do not understand the procedure, clarify with the concerned staff.
- Report any problem in system (if any) to the person in-charge.
- After the lab session, shut down the computers.
- All students in the laboratory should follow the directions given by staff/lab technical staff.

DON'TS

- Do not insert metal objects such as pins, needle or clips into the computer casing. They may cause fire.
- Do not open any irrelevant websites in labs.
- Do not use flash drive on laboratory computers without the consent of lab instructor.
- Do not upload, delete or alter any software/ system files on laboratory computers.
- Students are not allowed to work in laboratory alone or without presence of the teaching staff/ instructor.
- Do not change the system settings and keyboard keys.
- Do not damage any hardware.

DATA STRUCTURES LABORATORY – 17CSL38

LIST OF EXPERIMENTS

Laboratory Experiments:

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

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 Replace String (**REP**)
b. Perform Pattern Matching Operation: Find and Replace all occurrences of **PAT** in **STR** with **REP** if **PAT** exists in **STR.** Report suitable 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.

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**)
a. ***Push*** an Element on to Stack
b. ***Pop*** an Element from Stack
c. Demonstrate how Stack can be used to check ***Palindrome***
d. Demonstrate ***Overflow*** and ***Underflow*** situations on Stack
e. Display the status of Stack
f. Exit
Support the program with appropriate functions for each of the above operations

4. Design, Develop and Implement a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: **+, -, *, /, %(Remainder), ^(Power)** and **alphanumeric** operands**.**

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

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 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

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*** 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
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*** 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
9. Design, Develop and Implement a Program in C for the following operationson **Singly Circular Linked List (SCLL)** with header nodes a. Represent and Evaluate a Polynomial **P(x,y,z) = 6x2y2z-4yz5+3x3yz+2xy5z-2xyz3** 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
10. 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 Post Order c. Search the BST for a given element (**KEY**) and report the appropriate message e. Exit
11. 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
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. Design and develop a Program in C that uses Hash function **H: K ®L** as H(**K**)=**K** mod **m (remainder** method), and implement hashing technique to map a given key **K** to the address space **L.** Resolve the collision (if any) using **linear probing**.

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Belawadi, Srirangapatna Taluk, Mandya-571438

Department of Computer Science and Engineering

Academic Year: 2018-19 **Date: 27/07/2018**

Course Outcome

Subject: **Data Structures Laboratory** **Course Code: C238**
Subject Code: **17CSL38**

CO's	DESCRIPTION OF THE OUTCOMES
C238.1	Understand the behaviour of different data structures with their application and apply their suitability to solve a problem and documenting outcome of the experiments
C238.2	Apply data structure concepts and programming knowledge to solve real world problems.
C238.3	Analyze and evaluate the different data structures and verify the results.

CO No	PO No												PSO	
	1	2	3	4	5	6	7	8	9	10	11	12	1	2
C238.1	3	-	-	-	-	-	-	-	-	-	-	-	2	-
C238.2	-	-	3	-	2	-	-	-	-	-	-	-	2	2
C238.3	-	3	-	3	-	-	-	-	-	-	-	-	2	2
CO Average	**3**	**3**	**3**	**3**	**2**	-	-	-	-	-	-	-	**2**	**2**

01. Design, Develop and Implement a menu driven program in C for the following Array operations
a. Creating Array of N Integer elements.
b. Display of Array elements with suitable headings.
c. Inserting an element (ELEM) at a given valid position (POS).
d. Deleting an element at a given valid position (POS).
e. Exit.
Support the program with functions for each of the above operations.

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

int a[20], n, elem, i, pos;

void create()

{
printf("\nEnter the size of the array elements: ");
scanf("%d", &n);
printf("\nEnter the elements for the array:\n");
for(i=0; i<n; i++)
scanf("%d", &a[i]);
}

void display()
{
int i;
if(n>0)
{
printf("\nThe array elements are:\n");
for(i=0; i<n; i++)
printf("%d\t", a[i]);
}
else
printf("\n Array is empty\n");
}

void insert()
{
printf("\nEnter the position for the new element:
");
scanf("%d", &pos);
pos=pos-1;
if(pos<=n)
{
printf("\nEnter the element to be inserted: ");
scanf("%d", &elem);
for(i=n-1; i>=pos; i--)
a[i+1] = a[i];
a[pos] = elem;
n = n+1;
}
else
printf("\nEnter valid position\n");
}

void del()
{
printf("\nEnter the position of the element to be
deleted: ");
scanf("%d", &pos);
pos=pos-1;
if(pos<n && pos>=0)
{
elem = a[pos];
for(i=pos; i<n-1; i++)
{
a[i] = a[i+1];
}
n = n-1;
printf("\nThe deleted element is = %d", elem);
}
else
printf("\nEnter valid position\n");
}

void main()
{
int ch;
for(;;)
{
printf("\n\n--------Menu-----------\n");
printf("1.Create\n 2.Display\n 3.Insert\n
4.Delete\n 5.Exit\n");
printf("-------------------------------");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)
{
case 1: create();
break;
case 2: display();
break;
case 3: insert();
break;
case 4: del();
break;
case 5: exit(0);
break;
default: printf("\nInvalid choice:\n");
break;
}
}
```

```
}
```

OUTPUT:

```
--------Menu-----------
1.Create
 2.Display
 3.Insert
 4.Delete
 5.Exit
------------------------------
Enter your choice: 1

Enter the size of the array elements: 3

Enter the elements for the array:
1
2
3


--------Menu-----------
Enter your choice: 2

The array elements are:
1       2       3

--------Menu-----------
Enter your choice: 3

Enter the position for the new element: 2

Enter the element to be inserted: 22


--------Menu-----------
Enter your choice: 2

The array elements are:
1       22      2       3

--------Menu-----------

Enter your choice: 3

Enter the position for the new element: 1

Enter the element to be inserted: 11


--------Menu-----------
Enter your choice: 2

The array elements are:
11      1       22      2       3

--------Menu-----------
Enter your choice: 3

Enter the position for the new element: 5

Enter the element to be inserted: 55


--------Menu-----------
Enter your choice: 2

The array elements are:
11      1       22      2       55      3

--------Menu-----------
Enter your choice: 3

Enter the position for the new element: 7

Enter the element to be inserted: 77


--------Menu-----------
Enter your choice: 2

The array elements are:
11      1       22      2       55      3       77

--------Menu-----------
Enter your choice: 3

Enter the position for the new element: 9

Enter valid position


--------Menu-----------
Enter your choice: 3

Enter the position for the new element: 15

Enter valid position

--------Menu-----------
Enter your choice: 2

The array elements are:
11      1       22      2       55      3       77
```

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 1

The deleted element is = 11

--------Menu----------
Enter your choice: 2

The array elements are:
1 22 2 55 3 77

--------Menu----------
Enter your choice: 3

Enter the position for the new element: 4

Enter the element to be inserted: 44

--------Menu----------
Enter your choice: 2

The array elements are:
1 22 2 44 55 3 77

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 4

The deleted element is = 44

--------Menu----------
Enter your choice: 2

The array elements are:
1 22 2 55 3 77

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 4

The deleted element is = 55

--------Menu----------
Enter your choice: 2

The array elements are:
1 22 2 3 77

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 5

The deleted element is = 77

--------Menu----------
Enter your choice: 2

The array elements are:
1 22 2 3

--------Menu----------
Enter your choice: 3

Enter the position for the new element: 6

Enter valid position

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 9

Enter valid position

--------Menu----------
Enter your choice: 2

The array elements are:
1 22 2 3

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 2

The deleted element is = 22

--------Menu----------
Enter your choice: 2

The array elements are:
1 2 3

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 1

The deleted element is = 1

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 1

The deleted element is = 2

--------Menu----------
Enter your choice: 2

The array elements are:
3

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 1

The deleted element is = 3

--------Menu----------
Enter your choice: 2

Array is empty

--------Menu----------
Enter your choice: 1

Enter the size of the array elements: 3

Enter the elements for the array:
1
2
3

--------Menu----------
Enter your choice: 2

The array elements are:
1 2 3

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 7

Enter valid position

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 0

Enter valid position

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: -9

Enter valid position

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 0

Enter valid position

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 1

The deleted element is = 1

--------Menu----------
Enter your choice: 2

The array elements are:
2 3

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 1

The deleted element is = 2

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 1

The deleted element is = 3

--------Menu----------
Enter your choice: 4

Enter the position of the element to be deleted: 1

```
Enter valid position


--------Menu-----------
Enter your choice: 4

Enter the position of the element to be deleted: 1

Enter valid position


--------Menu-----------
Enter your choice: 4

Enter the position of the element to be deleted: 5

Enter valid position
```

---

02. Design, Develop and Implement a Program in C for the following operations on Strings
a. Read a main String (STR), a Pattern String (PAT) and a Replace String (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.

```
#include<stdio.h>
#include<stdlib.h>
char str[100],pat[20],rep[20];
int i,j,found=0,k,pos;
int slen=0,plen=0,rlen=0;

void read()
{
printf("enter main string :");
gets(str);
for(slen=0;str[slen]!='\0';slen++);
printf("enter pattern string:");
gets(pat);
for(plen=0;pat[plen]!='\0';plen++);
printf("enter replace string:");
gets(rep);
for(rlen=0;rep[rlen]!='\0';rlen++);

if(slen<plen || slen<plen)
{
printf("\nstring length is lesser matching cannot be done\n");
exit(0);
}
if(plen!=rlen)
{
printf("\npattern and replace string lengths are not equal, matching cannot be done\n");
exit(0);
}
printf("\nmain string is:%s",str);
printf("\npattern string is:%s",pat);
printf("\nreplace string is:%s",rep);
}

void matching()
{
i=j=0;
while(str[i]!='\0')
{
        j=0;
        pos=i;
        while(pat[j]!='\0')
                {
                if(str[i]==pat[j])
                        {
                                i++;
                                j++;
                        }
                else
                        {
                                if(str[i]==pat[0])
                                break;

                                else
                                {
                                i++;
                                break;
                                }
                        }
                }

        if(pat[j]=='\0')
                {

        for(k=pos,j=0;rep[j]!='\0';j++,k++)
                                str[k]=rep[j];
                        found=1;
                }
}}
```

```
void main()
{
read();
matching();
if(found==1)
{
printf("\npattern string found\n");
printf("\nString after replacement= %s\n",str);
}
else
printf("\npattern string not found\n");
}
```

OUTPUT 1:

```
enter main string :mit is in mysore, mit is good
enter pattern string:mit
enter replace string:MIT

main string is:mit is in mysore, mit is good
pattern string is:mit
replace string is:MIT
pattern string found

String after replacement= MIT is in mysore, MIT
is good
```

OUTPUT 2:

```
enter main string :ABCDABCDEF ABC ABC
ABC ABABC AABC AAAABCCCC
enter pattern string:ABC
enter replace string:xyz

main string is:ABCDABCDEF ABC ABC ABC
ABABC AABC AAAABCCCC
pattern string is:ABC
replace string is:xyz
pattern string found

String after replacement= xyzDxyzDEF xyz xyz
xyz ABxyz Axyz AAAxyzCCC
```

---

03. Design, Develop and Implement a menu driven program in C for the following operations on
STACK of integers (Array implementation of stack with maximum size MAX)
a. Push an element on to stack
b. Pop an element from stack.
c. Demonstrate how stack can be used to check palindrome.
d. Demonstrate Overflow and Underflow situations on stack.
e. Display the status of stack.
f. Exit.
Support the program with appropriate functions for each of the above operations.

```
#include<stdio.h>
#include<stdlib.h>
#define max 5
int ele,del,ch,top=-1,i;
int stack[max];

void push()
{

if(top==max-1)
printf("stack OVERFLOW\n");

else
{
printf("enter a element\n");
scanf("%d",&ele);
stack[++top]=ele;
}
}

void pop()
{
del=stack[top--];
}

void palindrome()
{
int pal=0,temp;
temp=top;
for(i=0;i<=top/2;i++)
{
pop();
if(stack[i]==del)
pal=1;
else
pal=0;
}
```

```
top=temp;
if(pal==1)
printf("STACK content is a palindrome\n");
else
printf("STACK content is NOT a palindrome\n");
}
void display()
{
if(top==-1)
{
printf("stack EMPTY\n");
return;
}
printf("stack ELEMENTS are:\n");
for(i=top;i>=0;i--)
printf("%d\n",stack[i]);
}



void main()
{

for(;;)
{
printf("\n----MENU----\n
1.push\n2.pop\n3.Palindrome\n4.display\n5.exit\n
enter your choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1: push();break;
case 2: if(top==-1)
        printf("stack UNDERFLOW\n");
        else
        {
        pop();
        printf("popped =%d\n",del);
        }
        break;
case 3: if(top==-1)
        printf("stack EMPTY, Palindrome cannot be
checked\n");
        else
        palindrome();
        break;
case 4: display(); break;
case 5: exit(0);
default: printf("invalid choice\n");
}
}
}
```

OUTPUT 1:

```
----MENU----
1.push
2.pop
3.Palindrome
4.display
5.exit
enter your choice
3
stack EMPTY, Palindrome cannot be checked

----MENU----
enter your choice
4
stack EMPTY
```

OUTPUT 2:

```
----MENU----
1.push
2.pop
3.Palindrome
4.display
5.exit
enter your choice
1
enter a element
1

----MENU----
enter your choice
1
enter a element
2

----MENU----
enter your choice
1
enter a element
3

----MENU----
enter your choice
4
stack ELEMENTS are:
3
2
1

----MENU----
enter your choice
1
enter a element
4
```

```
----MENU----
enter your choice
1
enter a element
5

----MENU----
enter your choice
1
stack OVERFLOW

----MENU----
enter your choice
4
stack ELEMENTS are:
5
4
3
2
1

----MENU----
enter your choice
3
STACK content is NOT a palindrome

----MENU----
enter your choice
2
popped =5

----MENU----
enter your choice
2
popped =4

----MENU----
enter your choice
2
popped =3

----MENU----
enter your choice
4
stack ELEMENTS are:
2
1

----MENU----
enter your choice
1
enter a element
1

----MENU----
enter your choice
3
STACK content is a palindrome

----MENU----
enter your choice
4
stack ELEMENTS are:
1
2
1
```

04: Design, Develop and Implement a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, %(Remainder), ^ (Power) and alphanumeric operands.

```
#include<stdio.h>
#define SIZE 50          /* Size of Stack */
#include <ctype.h>
char s[SIZE];
int top=-1;      /* Global declarations */

void 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 '/': return 3;
   case '%':
   case '^': return 4;
   }
}
```

```
void main()
{                       /* Main Program */
   char infx[50],pofx[50],ch,elem;
   int i=0,k=0;
   printf("\n\nEnter the Infix Expression:");
   scanf("%s",infx);
   push('#');
   while( (ch=infx[i++]) != '\0')
   {
      if( ch == '(') push(ch);
      else if(isalnum(ch)) pofx[k++]=ch;
      else if( ch == ')')
            {
               while( s[top] != '(')
                  pofx[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 Expression: %s  \n
Postfix Expression: %s\n",infx,pofx);
}
```

OUTPUT 1:

```
Enter the Infix Expression:a+b*(c/d)^e-f


Given Infix Expression: a+b*(c/d)^e-f
 Postfix Expression: abcd/e^*+f-
```

OUTPUT 2:

```
Enter the Infix Expression:1*3/(5-6%3)+4/5


Given Infix Expression: 1*3/(5-6%3)+4/5
 Postfix Expression: 13*563%-/45/+
```

---

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

```
#include<stdio.h>
#include <stdlib.h>
#include<string.h>
#include<math.h>
#include<ctype.h>
#define MAX 50           /* Size of Stack */

char post[50];
int s[MAX],op1,op2,res;
int top=-1;       /* Global declarations */

void push(int num)
{                    /* Function for PUSH operation */
   s[++top]=num;
}

int pop()
{                    /* Function for POP operation */
   return(s[top--]);
}

int eva(char ch,int op1,int op2)
{                /* Function for precedence */
   switch(ch)
   {

   case '+': return op1+op2;

   case '-': return op1-op2;

   case '*': return op1*op2;

   case '/': if(op2==0)
              {
                    printf("arthimatic error, result
          cannot be computed\n\n");
                 exit(0);
              }
             else
              return op1/op2;

   case '%':
          if(op2==0)
             {
                        printf("modulus error, result
          cannot be computed\n\n");
                 exit(0);
```

```
            }
          return op1%op2;

   case '^': return pow(op1,op2);

   default: printf("Invalid operator\n\n");
            exit(0);

   }
}

void main()
{                       /* Main Program */

   int i=0;
   char ch;
   printf("\n\nEnter the Postfix Expression:");
   scanf("%s",post);

   while( (ch=post[i++]) != '\0')
   {
      if(isalpha(ch))
      {
      printf("invalid expression\n");
      exit(0);
      }

      if(isdigit(ch))
            push(ch-'0');

          else
         {
         op2=pop();
         op1=pop();
         res=eva(ch,op1,op2);
         s[++top]=res;
}
}

printf("\n\n Postfix Expression: %s\n",post);
printf("\n\n Result: %d\n",s[top]);
}
```

OUTPUT 1:

```
Enter the Postfix Expression:1234+-*

 Postfix Expression: 1234+-*

 Result: -5
----------------------------------------
```

OUTPUT 2:

```
Enter the Postfix Expression:1234+++

 Postfix Expression: 1234+++

 Result: 10
----------------------------------------
```

OUTPUT 3:

```
Enter the Postfix Expression:13*563%-/45/+

 Postfix Expression: 13*563%-/45/+

 Result: 0
----------------------------------------
```

OUTPUT 4:

```
Enter the Postfix Expression:10/
arthimatic error, result cannot be computed
----------------------------------------
Enter the Postfix Expression:123*&
Invalid operator
----------------------------------------
Enter the Postfix Expression:123+-bc+
invalid expression
----------------------------------------
Enter the Postfix Expression:95%

 Postfix Expression: 95%

 Result: 4
------------------------------------------------------------
------------------------------------------------------------
```

Solving Tower of Hanoi problem with N disks.

```
#include<stdio.h>
#include<math.h>
void toh(int n, char s,char t, char d)
{
if(n==0)
return;
else
{
toh(n-1, s, d, t);
printf("\nMove disc %d from %c to %c\n", n, s,
d);
toh(n-1, t, s, d);
}
}

void main()
{
int n;
printf("\nEnter the number of discs: \n");
scanf("%d", &n);
toh(n, 'S', 'T', 'D');
```

```
printf("\n\nTotal Number of moves are: %d\n\n",
(int)pow(2,n)-1);
}
```

OUTPUT:

```
Enter the number of discs:
3

Move disc 1 from S to D
Move disc 2 from S to T
Move disc 1 from D to T
Move disc 3 from S to D
Move disc 1 from T to S
Move disc 2 from T to D
Move disc 1 from S to D

Total Number of moves are: 7
```

---

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

```
#include<stdio.h>
#include<stdlib.h>
#define max 5
char cq[max];
int f=-1,r=-1,ch;
char ele,del;

void insert()
{

if((f==0 && r==max-1)||(f==r+1))
printf("CIRCULAR-queue overflow\n");
else
{
getchar();
printf("enter a element:\n");
scanf("%c",&ele);
if(r==-1)
f=0;

r=(r+1)%max;
cq[r]=ele;
}
}

void delete()
{
if(f==-1&&r==-1)
printf("CIRCULAR-queue UNDERFLOW\n");

else
{
del=cq[f];
printf("deleted element:%c\n",del);
if(f==r)
f=r=-1;
else
f=(f+1)%max;
}
}

void display()
{
int i;
if(f==-1)
printf("CIRCULAR-queue EMPTY\n");
else
{
i=f;
printf("CIRCULAR-queue elements are:\n");
for(;i!=r;i=(i+1)%max)
printf("%d-->%c\n",i,cq[i]);
printf("%d-->%c\n",i,cq[r]);
}
}

void main()
{
for(;;)
{
printf("\n----MENU----\n
1.insert\n2.delete\n3.display\n4.exit\n enter your
choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1: insert();break;
case 2: delete(); break;
case 3: display(); break;
case 4: exit(0);}}}
```

OUTPUT:

```
1.insert
2.delete
3.display
4.exit
enter your choice
1
enter a element:
A

----MENU----
enter your choice
1
enter a element:
B

----MENU----
enter your choice
1
enter a element:
C

----MENU----
enter your choice
1
enter a element:
D

----MENU----
enter your choice
1
enter a element:
E

----MENU----
enter your choice
1
CIRCULAR-queue overflow

----MENU----
enter your choice
3
CIRCULAR-queue elements are:
0-->A
1-->B
2-->C
3-->D
4-->E

----MENU----
enter your choice
2
deleted element:A

----MENU----
enter your choice
2
deleted element:B

----MENU----
enter your choice
3
CIRCULAR-queue elements are:
2-->C
3-->D
4-->E

----MENU----
enter your choice
1
enter a element:
W

----MENU----
enter your choice
1
enter a element:
T

----MENU----

enter your choice
3
CIRCULAR-queue elements are:
2-->C
3-->D
4-->E
0-->W
1-->T

----MENU----
enter your choice
2
deleted element:C

----MENU----
enter your choice
2
deleted element:D

----MENU----
enter your choice
2
deleted element:E

----MENU----
enter your choice
```

```
2
deleted element:W

----MENU----
enter your choice
2
deleted element:T

----MENU----
enter your choice
2
CIRCULAR-queue UNDERFLOW

----MENU----
enter your choice
3
CIRCULAR-queue EMPTY
```

---

07. 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

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

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define max 10
typedef struct students
{
char usn[20],name[20],branch[10],phno[15];
int sem;
struct student *next;
}student;
student *head=NULL,*tail=NULL;

int ch,c=0;

void finsert()
{
char u[20],n[20],b[10],p[20];
int s;
if(c==max)
printf("student record full\n");
else
{
student *newstu=malloc(sizeof(student));
printf("enter student usn:");
scanf("%s",u);
printf("enter student name:");
scanf("%s",n);
printf("enter student branch:");
scanf("%s",b);
printf("enter student sem:");
scanf("%d",&s);
printf("enter student phno:");
scanf("%s",p);
strcpy(newstu->usn,u);
strcpy(newstu->name,n);
strcpy(newstu->branch,b);
newstu->sem=s;
strcpy(newstu->phno,p);
newstu->next=head;
head=newstu;
c++;
if(c==1)
tail=newstu;
}
}

void fdelete()
{
if(c==0)
printf("student record empty\n");
else
{
student *temp;
temp=head;
printf("deleted student record is:");
printf("usn:%s\n",temp->usn);
printf("name:%s\n",temp->name);
printf("branch:%s\n",temp->branch);
printf("sem:%d\n",temp->sem);
printf("phno:%s\n",temp->phno);
if(c==1)
head=tail=NULL;
else
head=head->next;
free(temp);
c--;
}
}

void rinsert()
{
char u[20],n[20],b[10],p[20];
int s;
if(c==max)
```

```
printf("student record full\n");
else
{
student *newstu=malloc(sizeof(student));
printf("enter student usn:");
scanf("%s",u);
printf("enter student name:");
scanf("%s",n);
printf("enter student branch:");
scanf("%s",b);
printf("enter student sem:");
scanf("%d",&s);
printf("enter student phno:");
scanf("%s",p);
strcpy(newstu->usn,u);
strcpy(newstu->name,n);
strcpy(newstu->branch,b);
newstu->sem=s;
strcpy(newstu->phno,p);
tail->next=newstu;
newstu->next=NULL;
tail=newstu;
c++;
}
}

void rdelete()
{
if(c==0)
printf("student record empty\n");
else
{
student *temp,*temp1;
temp=head;
temp1=tail;
printf("deleted student record is:\n");
printf("usn:%s\n",temp1->usn);
printf("name:%s\n",temp1->name);
printf("branch:%s\n",temp1->branch);
printf("sem:%d\n",temp1->sem);
printf("phno:%s\n",temp1->phno);
if(c==1)
head=tail=NULL;
else
{
while(temp->next!=tail)
temp=temp->next;
tail=temp;
tail->next=NULL;
}
free(temp1);
c--;
}
}

void display()
{
student * temp;
if(c==0)
{
printf("NO studentrecord to display\n");
return;
}
temp=head;
printf("number of student records are:%d\n",c);
printf("usn\tname\tbranch\tsem\tphno\n");
while(temp!=NULL)
{
printf("%s\t",temp->usn);
printf("%s\t",temp->name);
printf("%s\t",temp->branch);
printf("%d\t",temp->sem);
printf("%s\n",temp->phno);
temp=temp->next;
}
}

void create()
{
int n,i;
printf("enter how many students records:\n");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter student %d details:\n",i+1);
finsert();
}
}

void main()
{
for(;;)
{
printf("\n----MENU----\n 1.create from
front\n2.finsert\n3.fdelete\n4.rinsert\n5.rdelete\n6.
display\n7.exit\nenter your choice\n");
scanf("%d",&ch);
switch(ch)

{
case 1: create();
break;
case 2: finsert();
break;
case 3:fdelete();
break;
case 4: rinsert();
break;
case 5: rdelete();
```

```
break;
case 6: display();
break;
case 7: exit(0);
default: printf("Invalid choice\n");
}
}
}
```

OUTPUT:

```
1.create from front
2.finsert
3.fdelete
4.rinsert
5.rdelete
6.display
7.exit
enter your choice
1
enter how many students records:
3
enter student 1 details:
enter student usn:1
enter student name:Q
enter student branch:Q
enter student sem:1
enter student phno:1
enter student 2 details:
enter student usn:2
enter student name:W
enter student branch:W
enter student sem:2
enter student phno:2
enter student 3 details:
enter student usn:3
enter student name:E
enter student branch:E
enter student sem:3
enter student phno:3

----MENU----
enter your choice
6
number of student records are:3
usn     name    branch sem      phno
3       E       E       3       3
2       W       W       2       2
1       Q       Q       1       1

----MENU----
enter your choice
5
deleted student record is:
usn:1
name:Q
branch:Q
sem:1
phno:1

----MENU----
enter your choice
6
number of student records are:2
usn     name    branch sem      phno
3       E       E       3       3
2       W       W       2       2

----MENU----
enter your choice
3
deleted student record is:usn:3
name:E
branch:E
sem:3
phno:3

----MENU----
enter your choice
6
number of student records are:1
usn     name    branch sem      phno
2       W       W       2       2

----MENU----
enter your choice
4
enter student usn:5
enter student name:t
enter student branch:t
enter student sem:5
enter student phno:5

----MENU----
enter your choice
6
number of student records are:2
usn     name    branch sem      phno
2       W       W       2       2
5       t       t       5       5

----MENU----
enter your choice
5
deleted student record is:
usn:5
name:t
branch:t
```

```
sem:5
phno:5

----MENU----
enter your choice
5
deleted student record is:
usn:2
name:W
branch:W
sem:2
phno:2

----MENU----
enter your choice
5
student record empty

----MENU----
enter your choice
3
student record empty

----MENU----
enter your choice
6
NO student record to display
```

---

08. 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.
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
emp. Demonstrate how this DLL can be used as Double Ended Queue
f. Exit

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define max 5
typedef struct employee
{
char ssn[20],name[20],dept[10],des[10],phno[15];
int sal;
struct employee *next;
struct employee *pre;
}emp;

emp *head=NULL,*tail=NULL;

int ch,c=0;

emp * createemp()
{
char u[20],n[20],b[10],d[10],p[20];
int s;
printf("Front Insert\n");
emp *newemp=malloc(sizeof( emp));
printf("enter emp ssn:");
scanf("%s",u);
printf("enter emp name:");
scanf("%s",n);
printf("enter emp dept:");
scanf("%s",b);
printf("enter emp designation:");
scanf("%s",d);
printf("enter emp sal:");
scanf("%d",&s);
printf("enter emp phno:");
scanf("%s",p);
strcpy(newemp->ssn,u);
strcpy(newemp->name,n);
strcpy(newemp->dept,b);
strcpy(newemp->des,d);
newemp->sal=s;
strcpy(newemp->phno,p);
return newemp;
}

void deleteemp(emp *temp)
{
printf("deleted emp record is:");
printf("ssn:%s\n",temp->ssn);
printf("name:%s\n",temp->name);
printf("dept:%s\n",temp->dept);
printf("dept:%s\n",temp->des);
printf("sal:%d\n",temp->sal);
printf("phno:%s\n",temp->phno);
}

void finsert()
{

if(c==max)
printf("emp record full\n");

else
```

```
{
emp *newemp=NULL;
newemp=createemp();
if(c==0)
tail=newemp;
else
{
head->pre=newemp;
newemp->next=head;
newemp->pre=NULL;
}
head=newemp;
c++;
}
}

void fdelete()
{
if(c==0)
printf("emp record empty\n");
else
{
printf("Front Delete\n");
emp *temp;
temp=head;
deleteemp(temp);

if(c==1)
head=tail=NULL;
else
{
head=head->next;
head->pre=NULL;
}
free(temp);
c--;
}
}

void rinsert()
{
if(c==max)
printf("emp record full\n");
else
{
emp *newemp=NULL;
newemp=createemp();

if(c==0)
head=newemp;

else
{
tail->next=newemp;
newemp->pre=tail;
newemp->next=NULL;
}

tail=newemp;
c++;
}
}

void rdelete()
{
if(c==0)
printf("emp record empty\n");
else
{
printf("Rear Delete\n");
 emp *temp;
temp=tail;
deleteemp(temp);
if(c==1)
head=tail=NULL;
else
{
tail=tail->pre;
tail->next=NULL;
}
free(temp);
c--;
}
}

void display()
{
emp * temp;
if(c==0)
{
printf("NO emp record to display\n");
return;
}
temp=head;
printf("number of emp records are:%d\n",c);
printf("ssn\tname\tdept\tdes\tsal\tphno\n");
while(temp!=NULL)
{
printf("%s\t",temp->ssn);
printf("%s\t",temp->name);
printf("%s\t",temp->dept);
printf("%s\t",temp->des);
printf("%d\t",temp->sal);
printf("%s\n",temp->phno);
temp=temp->next;
}
}
```

```
void create()
{
int n,i;
printf("enter how many emps records:\n");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter emp %d details:\n",i+1);
rinsert();
}
}

void main()
{
for(;;)
{
printf("\n----MENU----\n 1.create from
rear\n2.finsert\n3.fdelete\n4.rinsert\n5.rdelete\n6.d
isplay\n7.exit\nenter your choice\n");
scanf("%d",&ch);
switch(ch)

{
case 1: create();
break;
case 2: finsert();
break;
case 3:fdelete();
break;
case 4: rinsert();
break;
case 5: rdelete();
break;
case 6: display();
break;
case 7: exit(0);
default: printf("Invalid choice\n");
}
}
}
```

OUTPUT:

```
1.create from rear
2.finsert
3.fdelete
4.rinsert
5.rdelete
6.display
7.exit
enter your choice
1
enter how many emps records:
1
enter emp 1 details:
Front Insert
enter emp ssn:1
enter emp name:1
enter emp dept:1
enter emp designation:1
enter emp sal:1
enter emp phno:1

----MENU----
enter your choice
6
number of emp records are:1
ssn     name    dept    des     sal     phno
1       1       1       1       1       1

----MENU----
enter your choice
4
Front Insert
enter emp ssn:2
enter emp name:2
enter emp dept:2
enter emp designation:2
enter emp sal:2
enter emp phno:2

----MENU----
enter your choice
6
number of emp records are:2
ssn     name    dept    des     sal     phno
1       1       1       1       1       1
2       2       2       2       2       2

----MENU----
enter your choice
2
Front Insert
enter emp ssn:3
enter emp name:3
enter emp dept:3
enter emp designation:3
enter emp sal:3
enter emp phno:3

----MENU----
enter your choice
6
number of emp records are:3
ssn     name    dept    des     sal     phno
3       3       3       3       3       3
1       1       1       1       1       1
```

2 2 2 2 2 2

----MENU----
enter your choice
4
Front Insert
enter emp ssn:4
enter emp name:4
enter emp dept:4
enter emp designation:4
enter emp sal:4
enter emp phno:4

----MENU----
enter your choice
6
number of emp records are:4

ssn	name	dept	des	sal	phno
3	3	3	3	3	3
1	1	1	1	1	1
2	2	2	2	2	2
4	4	4	4	4	4

----MENU----
enter your choice
4
Front Insert
enter emp ssn:5
enter emp name:5
enter emp dept:5
enter emp designation:5
enter emp sal:5
enter emp phno:5

----MENU----
enter your choice
6
number of emp records are:5

ssn	name	dept	des	sal	phno
3	3	3	3	3	3
1	1	1	1	1	1
2	2	2	2	2	2
4	4	4	4	4	4
5	5	5	5	5	5

----MENU----
enter your choice
4
emp record full

----MENU----
enter your choice
2
emp record full

----MENU----
enter your choice
1
enter how many emps records:
1
enter emp 1 details:
emp record full

----MENU----
enter your choice
6
number of emp records are:5

ssn	name	dept	des	sal	phno
3	3	3	3	3	3
1	1	1	1	1	1
2	2	2	2	2	2
4	4	4	4	4	4
5	5	5	5	5	5

----MENU----
enter your choice
5
Rear Delete
deleted emp record is:ssn:5
name:5
dept:5
dept:5
sal:5
phno:5

----MENU----
enter your choice
5
Rear Delete
deleted emp record is:ssn:4
name:4
dept:4
dept:4
sal:4
phno:4

----MENU----
enter your choice
6
number of emp records are:3

ssn	name	dept	des	sal	phno
3	3	3	3	3	3
1	1	1	1	1	1
2	2	2	2	2	2

----MENU----
enter your choice
3

```
Front Delete
deleted emp record is:ssn:3
name:3
dept:3
dept:3
sal:3
phno:3

----MENU----
enter your choice
6
number of emp records are:2
ssn     name    dept    des     sal     phno
1       1       1       1       1       1
2       2       2       2       2       2

----MENU----
enter your choice
3
Front Delete
deleted emp record is:ssn:1
name:1
dept:1
dept:1
sal:1
phno:1

----MENU----
enter your choice
3
Front Delete
deleted emp record is:ssn:2
name:2
dept:2
dept:2
sal:2
phno:2

----MENU----
enter your choice
3
emp record empty

----MENU----
enter your choice
5
emp record empty

----MENU----
enter your choice
6
NO emp record to display
------------------------------------------------------------
------------------------------------------------------------
```

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

a. Represent and Evaluate a Polynomial P(x,y,z) = 6x2y2z-4yz5+3x3yz+2xy5z-2xyz3

b. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z)

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

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

typedef struct polynomial
{
int pcoff,px,py,pz,flag;
struct polynomial *next;
}POLY;


POLY* getnode()
{
POLY *x;
x=(POLY*)malloc(sizeof(POLY));
if(x==NULL)
{
printf("Insufficient memory\n"); exit(0);
}
return x;
}

POLY* rinsert(int cf,int x,int y,int z,POLY *
head)
{
POLY *cur;

POLY * newterm=malloc(sizeof(POLY));
newterm->pcoff=cf;
newterm->px=x;
newterm->py=y;
newterm->pz=z;

cur=head->next;
while(cur->next!=head)
cur=cur->next;

cur->next=newterm;
newterm->next=head;
```

```
return head;
}

POLY* read_poly(POLY *head)
{
int x,y,z,cf,i,nterm;
printf("\nEnter no. of terms in polynomial: ");
scanf("%d",&nterm);
for(i=0;i<nterm;i++)
{
printf("\nEnter cof,x, y, z powers(0-indiacate NO
term): ");
scanf("%d%d%d%d",&cf,&x,&y,&z);
head=rinsert(cf,x,y,z,head);
}
return head;
}

void display(POLY *head)
{
POLY *temp;

if(head->next==head)
{
printf("Polynomial does not exist\n");
return;
}
temp=head->next;

printf("\nPolynomial is:\n");
while(temp!=head)
{
printf("%d x^%d y^%d z^%d",temp->pcoff,temp-
>px,temp->py,temp->pz);
if(temp->next != head)
printf(" + ");
temp=temp->next;
}
printf("\n");
}

void evaluate(POLY *h1)
{
POLY *temp;
int x=0, y=0, z=0,c=0;
float result=0;
temp=h1->next;
printf("\nEnter x, y, z, terms to evaluate:\n");
scanf("%d%d%d", &x, &y, &z);

while(temp!=h1)
{
result = result + (temp->pcoff * pow(x,temp->px)
* pow(y,temp->py) * pow(z,temp->pz));
temp=temp->next;
c++;
}
printf("\nPolynomial result is: %f", result);
}


POLY * add_poly(POLY *h1,POLY *h2,POLY
*h3)
{
POLY *p1,*p2;
int x1,x2,y1,y2,z1,z2,cf1,cf2,cf;
p1=h1->next;
while(p1!=h1)
{
x1=p1->px;
y1=p1->py;
z1=p1->pz;
cf1=p1->pcoff;
p2=h2->next;
        while(p2!=h2)
        {
        x2=p2->px;
        y2=p2->py;
        z2=p2->pz;
        cf2=p2->pcoff;
        if(x1==x2 && y1==y2 && z1==z2)
        break;
        p2=p2->next;
        }
if(p2!=h2)
{
cf=cf1+cf2;
p2->flag=1;
if(cf!=0)
h3=rinsert(cf,x1,y1,z1,h3);
}
else
h3=rinsert(cf1,x1,y1,z1,h3);
p1=p1->next;
}

p2=h2->next;
while(p2!=h2)
{
if(p2->flag==0)
h3=rinsert(p2->pcoff,p2->px,p2->py,p2->pz,h3);
p2=p2->next;
}
return h3;
}
```

```
void main()
{
int ch;
POLY *h1,*h2,*h3,*res;
h1=getnode();
h2=getnode();
h3=getnode();
res=getnode();
h1->next=h1;
h2->next=h2;
h3->next=h3;
res->next=res;

for(;;)
{
printf("\n1.Evaluate poly\n2.add 2
polynomials\n3.exit\nenter your choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("\nEnter polynomial to evaluate:\n");
h1=read_poly(h1);
display(h1);
evaluate(h1);
break;

case 2:
printf("\nEnter the first polynomial:");
h2=read_poly(h2);
printf("\nEnter the second polynomial:");
h3=read_poly(h3);
res=add_poly(h2,h3,res);
printf("\nFirst polynomial is: ");
display(h2);
printf("\nSecond polynomial is: ");
display(h3);
printf("\nThe sum of 2 polynomials is: ");
display(res);
break;

case 3: exit(0);
default: printf("Invalid choice\n");
}
}
}
```

OUTPUT:

```
1.Evaluate poly
2.add 2 polynomials
3.exit
enter your choice
1

Enter polynomial to evaluate:

Enter no. of terms in polynomial: 3

Enter cof,x, y, z powers(0-indiacate NO term): 4
1
2
3

Enter cof,x, y, z powers(0-indiacate NO term): 5
4
3
2

Enter cof,x, y, z powers(0-indiacate NO term): -4
3
3
3

Polynomial is:
4 x^1 y^2 z^3 + 5 x^4 y^3 z^2 + -4 x^3 y^3 z^3

Enter x, y, z, terms to evaluate:
1
2
1

Polynomial result is: 24.000000
------------------------------------------------------------

1.Evaluate poly
2.add 2 polynomials
3.exit
enter your choice
2

Enter the first polynomial:
Enter no. of terms in polynomial: 3

Enter cof,x, y, z powers(0-indiacate NO term): 5
4
3
2

Enter cof,x, y, z powers(0-indiacate NO term): 7
6
5
4

Enter cof,x, y, z powers(0-indiacate NO term): -9
3
2
1
```

```
Enter the second polynomial:
Enter no. of terms in polynomial: 4

Enter cof,x, y, z powers(0-indiacate NO term): 4
4
3
2

Enter cof,x, y, z powers(0-indiacate NO term): 2
6
5
4

Enter cof,x, y, z powers(0-indiacate NO term): 8
5
5
5

Enter cof,x, y, z powers(0-indiacate NO term): 2
3
2
1

First polynomial is:
Polynomial is:
5 x^4 y^3 z^2 + 7 x^6 y^5 z^4 + -9 x^3 y^2 z^1

Second polynomial is:
Polynomial is:
4 x^4 y^3 z^2 + 2 x^6 y^5 z^4 + 8 x^5 y^5 z^5 +
2 x^3 y^2 z^1

The sum of 2 polynomials is:
Polynomial is:
9 x^4 y^3 z^2 + 9 x^6 y^5 z^4 + -7 x^3 y^2 z^1 +
8 x^5 y^5 z^5

------------------------------------------------------------
------------------------------------------------------------
```

10. Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (node) of Integers
a. Create a node of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2
b. Traverse the node in Inorder, Preorder and Post Order
c. Search the node for a given element (KEY) and report the appropriate message
d. Exit

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

struct node
{
int data;
struct node *left;
struct node *right;
};
typedef struct node BST;
BST *root=NULL;

BST* createtree(BST *root, int data)
{
if (root == NULL)
{
BST *newnode=malloc(sizeof(BST));
newnode->data = data;
newnode->left = newnode->right = NULL;
return newnode;
}

if (data < (root->data))
root->left = createtree(root->left, data);

else if (data > root->data)
root -> right = createtree(root->right, data);

return root;
}

BST* search(BST *root, int data)
{
if(root == NULL)
printf("\nElement not found");
else if(data < root->data)
root->left=search(root->left, data);

else if(data > root->data)
root->right=search(root->right, data);

else
```

```
printf("\nElement found is: %d", root->data);

return root;
}

void inorder(BST *root)
{
if(root != NULL)
{
inorder(root->left);
printf("%d\t", root->data);
inorder(root->right);
}
}

void preorder(BST *root)
{
if(root != NULL)
{
printf("%d\t", root->data);
preorder(root->left);
preorder(root->right);
}
}

void postorder(BST *root)
{
if(root != NULL)
{
postorder(root->left);
postorder(root->right);
printf("%d\t", root->data);
}
}

void main()
{
int data, ch, i, n;

//BST *root=NULL;

while (1)
{
printf("\n1.Insertion in Binary Search Tree");
printf("\n2.Search Element in Binary Search
Tree");
printf("\n3.Inorder\n4.Preorder\n5.Postorder\n6.E
xit");
printf("\nEnter your choice: ");
scanf("%d", &ch);

switch (ch)
{
case 1: printf("\nEnter N value: " );
scanf("%d", &n);
printf("\nEnter the values to create BST
like(6,9,5,2,8,15,24,14,7,8,5,2)\n");
for(i=0; i<n; i++)
{
scanf("%d", &data);
root=createtree(root, data);
}
break;

case 2: printf("\nEnter the element to search: ");
scanf("%d", &data);
root=search(root, data);
break;

case 3: printf("\nInorder Traversal: \n");
inorder(root);
break;
case 4: printf("\nPreorder Traversal: \n");
preorder(root);
break;
case 5: printf("\nPostorder Traversal: \n");
postorder(root);
break;
case 6: exit(0);
default:printf("\nWrong option");
break;
}
}
}
```

OUTPUT:

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

Enter N value: 8

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

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

```
Inorder Traversal:
2  5    6       8       9       14      15      24

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

Preorder Traversal:
6  5    2       9       8       15      14      24

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

Postorder Traversal:
2  5    8       14      24      15      9       6

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

Enter the element to search: 8

Element found is: 8
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
3.Inorder
4.Preorder
5.Postorder
6.Exit
Enter your choice: 2

Enter the element to search: 5

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

Enter the element to search: 25

Element not found

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

Enter the element to search: 24

Element found is: 24
```

---

11. 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 BFS method

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

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

int a[10][10], n, source, b[10]={0};

void create()
{
int i,j;
printf("\nEnter the number of vertices of the
digraph: ");
scanf("%d", &n);
printf("\nEnter the adjacency matrix of the
graph:\n");
for(i=0; i<n; i++)
for(j=0; j<n; j++)
scanf("%d", &a[i][j]);

printf("\nthe adjacency matrix of the graph:\n");
for(i=0; i<n; i++)
{
for(j=0; j<n; j++)
printf("%d  ", a[i][j]);
printf("\n");
```

```
}
}

void bfs()
{
int q[10], u, front=0, rear=-1,i;
int visited[10]={0};
printf("\nEnter the source vertex to find other
nodes reachable or not: ");
scanf("%d", &source);
q[++rear] = source;
visited[source] = 1;

printf("\nThe reachable vertices are: ");
while(front<=rear)
{
u = q[front++];
for(i=0; i<n; i++)
{
if(a[u][i] == 1 && visited[i] == 0)
{
q[++rear] = i;
visited[i] = 1;
printf("\n%d-->%d",u,i);
}
}
}

for(i=0;i<n;i++)
{
if(i==source)
continue;
if(visited[i]==1)
printf("\nThe vertex %d is reachable from source
node %d\n",i,source);
else
printf("\nThe vertex %d is  not reachable from
source node %d\n",i,source);
}
}

void dfs(int source)
{
int i, top = -1;
b[source] = 1;
for(i=0; i<n; i++)
{
if(a[source][i] == 1 && b[i] == 0)
{
printf("\n%d -> %d", source, i);
dfs(i);
}
}
}
```

```
void main()
{
int ch,i;
while(1)
{
printf("\n1.Create
Graph\n2.BFS\n3.DFS\n4.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)
{
case 1: create(); break;

case 2: bfs();  break;

case 3: printf("\nEnter the source vertex: ");
scanf("%d",&source);
for(i=0;i<n;i++)
b[i]=0;

dfs(source);

for(i=0;i<n;i++)
{
if(i==source)
continue;
if(b[i]==1)
printf("\nThe vertex %d is reachable from source
node %d\n",i,source);
else
printf("\nThe vertex %d is  not reachable from
source node %d\n",i,source);
}

break;
case 4:exit(0);

default: printf("enter a valid choice\n");
}
}
}
```

OUTPUT 1:

```
1.Create Graph
2.BFS
3.DFS
4.Exit
Enter your choice: 1

Enter the number of vertices of the digraph: 6
```

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

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

1.Create Graph
2.BFS
3.DFS
4.Exit
Enter your choice: 2

Enter the source vertex to find other nodes
reachable or not: 0

The reachable vertices are:
0-->1
1-->2
2-->4
4-->3
3-->5
The vertex 1 is reachable from source node 0

The vertex 2 is reachable from source node 0

The vertex 3 is reachable from source node 0

The vertex 4 is reachable from source node 0

The vertex 5 is reachable from source node 0

1.Create Graph
2.BFS
3.DFS
4.Exit
Enter your choice: 2

Enter the source vertex to find other nodes
reachable or not: 4

The reachable vertices are:
4-->1
4-->3
1-->2
3-->5
5-->0
The vertex 0 is reachable from source node 4

The vertex 1 is reachable from source node 4

The vertex 2 is reachable from source node 4

The vertex 3 is reachable from source node 4

The vertex 5 is reachable from source node 4

1.Create Graph
2.BFS
3.DFS
4.Exit
Enter your choice: 3

Enter the source vertex: 5

5 -> 0
0 -> 1
1 -> 2
2 -> 4
4 -> 3
The vertex 0 is reachable from source node 5

The vertex 1 is reachable from source node 5

The vertex 2 is reachable from source node 5

The vertex 3 is reachable from source node 5

The vertex 4 is reachable from source node 5

1.Create Graph
2.BFS
3.DFS
4.Exit
Enter your choice: 3

Enter the source vertex: 3

3 -> 2
2 -> 4
4 -> 1
3 -> 5
5 -> 0
The vertex 0 is reachable from source node 3

The vertex 1 is reachable from source node 3

The vertex 2 is reachable from source node 3
```

```
The vertex 4 is reachable from source node 3

The vertex 5 is reachable from source node 3
-----------------------------------------------------
```

OUTPUT 2:

```
1.Create Graph
2.BFS
3.DFS
4.Exit
Enter your choice: 1

Enter the number of vertices of the digraph: 5

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

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

1.Create Graph
2.BFS
3.DFS
4.Exit
Enter your choice: 2

Enter the source vertex to find other nodes
reachable or not: 0

The reachable vertices are:
0-->1
0-->4
4-->2
The vertex 1 is reachable from source node 0

The vertex 2 is reachable from source node 0

The vertex 3 is  not reachable from source node 0

The vertex 4 is reachable from source node 0

1.Create Graph
2.BFS
3.DFS
4.Exit
Enter your choice: 2

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

The reachable vertices are:
1-->4
4-->2
The vertex 0 is  not reachable from source node 1

The vertex 2 is reachable from source node 1

The vertex 3 is  not reachable from source node 1

The vertex 4 is reachable from source node 1

1.Create Graph
2.BFS
3.DFS
4.Exit
Enter your choice: 3

Enter the source vertex: 0

0 -> 1
1 -> 4
4 -> 2
The vertex 1 is reachable from source node 0

The vertex 2 is reachable from source node 0

The vertex 3 is  not reachable from source node 0

The vertex 4 is reachable from source node 0

1.Create Graph
2.BFS
3.DFS
4.Exit
Enter your choice: 3

Enter the source vertex: 1

1 -> 4
4 -> 2
The vertex 0 is  not reachable from source node 1

The vertex 2 is reachable from source node 1

The vertex 3 is  not reachable from source node 1

The vertex 4 is reachable from source node 1

1.Create Graph
2.BFS
```

```
3.DFS
4.Exit
Enter your choice: 3

Enter the source vertex: 3

3 -> 2
2 -> 1
1 -> 4
The vertex 0 is  not reachable from source node 3

The vertex 1 is reachable from source node 3

The vertex 2 is reachable from source node 3

The vertex 4 is reachable from source node 3
```

---

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. Design and develop a Program in C that uses Hash function H: K →L as H(K)=K mod m (remainder method), and implement hashing technique to map a given key K to the address space L.
Resolve the collision (if any) using linear probing.

```
#include <stdio.h>
#include <stdlib.h>
#include<string.h>
#define MAX 10

struct employee
{
int id;
char name[15];
int k;
int flag;
}EMP;


struct employee emp[MAX];

int HT[MAX];
FILE *fptr;

void display()
{
int i;
printf("\nThe hash table is:\n");
printf("\n-----------------------------------------------
------");
printf("\nHTIndex\t KEY\t EmpID\t
EmpName\tCollision_Status");
printf("\n-----------------------------------------------
------\n");
for(i=0; i<MAX; i++)
printf("\n%d\t %d\t %d\t %10s\t %d",i,emp[i].k,
emp[i].id, emp[i].name,emp[i].flag);
printf("\n-----------------------------------------------
------\n");
}


void hashing()
{
int key,eid;
char ename[20];
int htindex,count=0;

while(fscanf(fptr,"%d%d%s",&key,&eid,ename)!
=EOF)
{
if(count == MAX)
{
printf("\n Hash table is full\na");
return;
}

htindex=key%MAX;

if(HT[htindex] == -1)
{
emp[htindex].id=eid;
strcpy(emp[htindex].name,ename);
emp[htindex].k=key;
emp[htindex].flag=0;
HT[htindex]=htindex;
}

else
{
printf("\nCollision Detected...!!!\n");

while(HT[htindex] != -1)
htindex=(htindex+1)%MAX;

emp[htindex].id=eid;
strcpy(emp[htindex].name,ename);
emp[htindex].k=key;
```

```
emp[htindex].flag=1;
HT[htindex]=htindex;

printf("\nCollision avoided successfully for
key=%d\n",key);
}
count++;
}
}

void main()
{
int i;
int ch;
fptr = fopen("a.txt","r");
printf("\nCollision handling by using LINEAR
PROBING: ");
for (i=0; i < MAX; i++)
HT[i] = -1;

for(;;)
{
printf("\n1.Create Employee record\n2.Display
Hash Table\n3.exit\nenter your choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1:
hashing();
break;

case 2:
display(emp);
break;

case 3: exit(0);

default: printf("invalid choice\n");
}
}
fclose(fptr);
}
```

OUTPUT:

FILE CONTENT

```
1234 111    anand
2345 222    kumar
3422 333    arun
5655 444    nayana
1112 555    eshwar
7876 666    rani
9988 777    manju
6544 888    thanu
```

```
Collision handling by using LINEAR PROBING:
1.Create Employee record
2.Display Hash Table
3.exit
enter your choice
1

Collision Detected...!!!

Collision avoided successfully for key=5655

Collision Detected...!!!

Collision avoided successfully for key=1112

Collision Detected...!!!

Collision avoided successfully for key=7876

Collision Detected...!!!

Collision avoided successfully for key=6544

1.Create Employee record
2.Display Hash Table
3.exit
enter your choice
2

The hash table is:
```

HTIndex	KEY	EmpID	EmpName	Collision Status
0	0	0	0	
1	0	0	0	
2	3422	333	arun	0
3	1112	555	eshwar	1
4	1234	111	anand	0
5	2345	222	kumar	0
6	5655	444	nayana	1
7	7876	666	rani	1
8	9988	777	manju	0
9	6544	888	thanu	1

Viva Question and Answers

1. What is a Data Structure?

A data structure is a way of organizing data that considers not only the items stored, but also their relationship to each other. Advance knowledge about the relationship between data items allows designing of efficient algorithms for the manipulation of data.

2. What are linear and non linear data Structures?

•Linear: A data structure is said to be linear if its elements form a sequence or a linear list. Examples: Array. Linked List, Stacks and Queues

•Non-Linear: A data structure is said to be non-linear if traversal of nodes is nonlinear in nature. Example: Graph and Trees

3. What is the data structures used to perform recursion?

Stack. Because of its LIFO (Last In First Out) property it remembers its 'caller' so knows whom to return when the function has to return. Recursion makes use of system stack for storing the return addresses of the function calls.

Every recursive function has its equivalent iterative (non-recursive) function. Even when such equivalent iterative procedures are written, explicit stack is to be used.

4. List out few of the Application of tree data-structure?

1. The manipulation of Arithmetic expression,
2. Symbol Table construction,
3. Syntax analysis.
4. Dynamic Memory Management
5. Disk scheduling
6. CPU scheduling

5. Differentiate between file and structure storage structure.

The key difference between both the data structure is the memory area that is being accessed. When dealing with the structure that resides the main memory of the computer system, this is referred to as storage structure. When dealing with an auxiliary structure, we refer to it as file structures.

6. How do you reference all the elements in a one-dimension array?

To reference all the elements in a one -dimension array, you need to use an indexed loop, So that, the counter runs from 0 to the array size minus one. In this manner, You can reference all the elements in sequence by using the loop counter as the array subscript.

7. In what areas do data structures are applied?

Data structures are essential in almost every aspect where data is involved. In general, algorithms that involve efficient data structure is applied in the following areas: numerical analysis, operating system, A.I., compiler design, database management, graphics, and statistical analysis, to name a few

8. What is LIFO?

LIFO is a short form of Last In First Out. It refers how data is accessed, stored and retrieved. Using this scheme, data that was stored last should be the one to be extracted first. This also means that in order to gain access to the first data, all the other data that was stored before this first data must first be retrieved and extracted.

9.What is a queue?

A queue is a data structure that can simulate a list or stream of data. In this structure, new elements are inserted at one end, and existing elements are removed from the other end.

10) Which data structures are applied when dealing with a recursive function?

Recursion, is a function that calls itself based on a terminating condition, makes use of the stack. Using LIFO, a call to a recursive function saves the return address so that it knows how to return to the calling function after the call terminates.

11) What is a stack?

A stack is a data structure in which only the top element can be accessed. As data is stored in the stack, each data is pushed downward, leaving the most recently added data on top.

12) Explain Binary Search Tree

A binary search tree stores data in such a way that they can be retrieved very efficiently. The left subtree contains nodes whose keys are less than the node's key value, while the right subtree contains nodes whose keys are greater than or equal to the node's key value. Moreover, both subtrees are also binary search trees.

13) What are multidimensional arrays?

Multidimensional arrays make use of multiple indexes to store data. It is useful when storing data that cannot be represented using single dimensional indexing, such as data representation in a board game, tables with data stored in more than one column.

14) Are linked lists considered linear or non-linear data structures?

It depends on where you intend to apply linked lists. If you based it on storage, a linked list is considered non-linear. On the other hand, if you based it on access strategies, then a linked list is considered linear.

15) How does dynamic memory allocation help in managing data?

Apart from being able to store simple structured data types, dynamic memory allocation can combine separately allocated structured blocks to form composite structures that expand and contract as needed.

16) What is FIFO?

FIFO stands for First-in, First-out, and is used to represent how data is accessed in a queue. Data has been inserted into the queue list the longest is the one that is removed first.

17) What is an ordered list?

An ordered list is a list in which each node's position in the list is determined by the value of its key component, so that the key values form an increasing sequence, as the list is traversed.

18) What is merge sort?

Merge sort, is a divide-and-conquer approach for sorting the data. In a sequence of data, adjacent ones are merged and sorted to create bigger sorted lists. These sorted lists are then merged again to form an even bigger sorted list, which continues until you have one single sorted list.

19) Differentiate NULL and VOID

Null is a value, whereas Void is a data type identifier. A variable that is given a Null value indicates an empty value. The void is used to identify pointers as having no initial size.

20) What is the primary advantage of a linked list?

A linked list is an ideal data structure because it can be modified easily. This means that **editing a linked list works regardless of how many elements are in the list.**

21) What is the difference between a PUSH and a POP?

Pushing and popping applies to the way data is stored and retrieved in a stack. A push denotes data being added to it, meaning data is being "pushed" into the stack. On the other hand, a pop denotes data retrieval, and in particular, refers to the topmost data being accessed.

22) What is a linear search?

A linear search refers to the way a target key is being searched in a sequential data structure. In this method, each element in the list is checked and compared against the target key. The process is repeated until found or if the end of the file has been reached.

23) How does variable declaration affect memory allocation?

The amount of memory to be allocated or reserved would depend on the data type of the variable being declared. For example, if a variable is declared to be of integer type, then 32 bits of memory storage will be reserved for that variable.

24) What is the advantage of the heap over a stack?

The heap is more flexible than the stack. That's because memory space for the heap can be dynamically allocated and de-allocated as needed. However, the memory of the heap can at times be slower when compared to that stack.

25) What is a postfix expression?

A postfix expression is an expression in which each operator follows its operands. The advantage of this form is that there is no need to group sub-expressions in parentheses or to consider operator precedence.

26) What is Data abstraction?

Data abstraction is a powerful tool for breaking down complex data problems into manageable chunks. This is applied by initially specifying the data objects involved and the operations to be performed on these data objects without being overly concerned with how the data objects will be represented and stored in memory.

27) How do you insert a new item in a binary search tree?

Assuming that the data to be inserted is a unique value (that is, not an existing entry in the tree), check first if the tree is empty. If it's empty, just insert the new item in the root node. If it's not empty, refer to the new item's key. If it's smaller than the root's key, insert it into the root's left subtree, otherwise, insert it into the root's right subtree.

28) How does a selection sort work for an array?

The selection sort is a fairly intuitive sorting algorithm, though not necessarily efficient. In this process, the smallest element is first located and switched with the element at subscript zero, thereby placing the smallest element in the first position.

The smallest element remaining in the subarray is then located next to subscripts 1 through n-1 and switched with the element at subscript 1, thereby placing the second smallest element in the second position. The steps are repeated in the same manner till the last element.

29) How do signed and unsigned numbers affect memory?

In the case of signed numbers, the first bit is used to indicate whether positive or negative, which leaves you with one bit short. With unsigned numbers, you have all bits available for that number. The effect is best seen in the number range (an unsigned 8-bit number has a range 0-255, while the 8-bit signed number has a range -128 to +127.

30) What is the minimum number of nodes that a binary tree can have?

A binary tree can have a minimum of zero nodes, which occurs when the nodes have NULL values. Furthermore, a binary tree can also have 1 or 2 nodes.

31) What are dynamic data structures?

Dynamic data structures are structures that expand and contract as a program runs. It provides a flexible means of manipulating data because it can adjust according to the size of the data.

32) In what data structures are pointers applied?

Pointers that are used in linked list have various applications in the data structure. Data structures that make use of this concept include the Stack, Queue, Linked List and Binary Tree.

33) Do all declaration statements result in a fixed reservation in memory?

Most declarations do, with the exemption of pointers. Pointer declaration does not allocate memory for data, but for the address of the pointer variable. Actual memory allocation for the data comes during run-time.

34) What are ARRAYs?

When dealing with arrays, data is stored and retrieved using an index that refers to the element number in the data sequence. This means that data can be accessed in any order. In programming, an array is declared as a variable having a number of indexed elements.

35) What is the minimum number of queues needed when implementing a priority queue?

The minimum number of queues needed in this case is two. One queue is intended for sorting priorities while the other queue is used for actual storage of data.

36) Which sorting algorithm is considered the fastest?

There are many types of sorting algorithms: quick sort, bubble sort, balloon sort, radix sort, merge sort, etc. Not one can be considered the fastest because each algorithm is designed for a particular data structure and data set. It would depend on the data set that you would want to sort.

37) Differentiate STACK from ARRAY.

Stack follows a LIFO pattern. It means that data access follows a sequence wherein the last data to be stored when the first one to be extracted. Arrays, on the other hand, does not follow a particular order and instead can be accessed by referring to the indexed element within the array.

38) Give a basic algorithm for searching a binary search tree.

1.if the tree is empty, then the target is not in the tree, end search
2. if the tree is not empty, the target is in the tree
3. check if the target is in the root item
4. if a target is not in the root item, check if a target is smaller than the root's value
5. if a target is smaller than the root's value, search the left subtree
6. else, search the right subtree

39) What is a dequeue?

A dequeue is a double-ended queue. This is a structure wherein elements can be inserted or removed from either end.

40) What is a bubble sort and how do you perform it?

A bubble sort is one sorting technique that can be applied to data structures such as an array. It works by comparing adjacent elements and exchanges their values if they are out of order. This method lets the smaller values "bubble" to the top of the list, while the larger value sinks to the bottom.

41) What are the parts of a linked list?

A linked list typically has two parts: the head and the tail. Between the head and tail lie the actual nodes. All these nodes are linked sequentially.

42) How does selection sort work?

Selection sort works by picking the smallest number from the list and placing it at the front. This process is repeated for the second position towards the end of the list. It is the simplest sort algorithm.

43) What is a graph?

A graph is one type of data structure that contains a set of ordered pairs. These ordered pairs are also referred to as edges or arcs and are used to connect nodes where data can be stored and retrieved.

44) Differentiate linear from a nonlinear data structure.

The linear data structure is a structure wherein data elements are adjacent to each other. Examples of linear data structure include arrays, linked lists, stacks, and queues. On the other hand, a non-linear data structure is a structure wherein each data element can connect to more than two adjacent data elements. Examples of nonlinear data structure include trees and graphs.

45) What is an AVL tree?

An AVL tree is a type of binary search tree that is always in a state of partially balanced. The balance is measured as a difference between the heights of the subtrees from the root. This self-balancing tree was known to be the first data structure to be designed as such.

46) What are doubly linked lists?

Doubly linked lists are a special type of linked list wherein traversal across the data elements can be done in both directions. This is made possible by having two links in every node, one that links to the next node and another one that connects to the previous node.

47) What are binary trees?

A binary tree is one type of data structure that has two nodes, a left node, and a right node. In programming, binary trees are an extension of the linked list structures.

48) Briefly explain recursive algorithm.

Recursive algorithm targets a problem by dividing it into smaller, manageable sub-problems. The output of one recursion after processing one sub-problem becomes the input to the next recursive process.

49) How do you search for a target key in a linked list?

To find the target key in a linked list, you have to apply sequential search. Each node is traversed and compared with the target key, and if it is different, then it follows the link to the next node. This traversal continues until either the target key is found or if the last node is reached.

50) Which data structures are used for BFS and DFS of a graph?

•Queue is used for BFS

•Stack is used for DFS. DFS can also be implemented using recursion (Note that recursion also uses function call stack).