YENEPOYA
(DEEMED TO BE UNIVERSITY)

DEPARTMENT OF COMPUTER SCIENCE

DATA STRUCTURES AND ALGORITHMS LAB MANUAL 2024-25

(BCA/BSC)

SEMESTER	**:**	**3**	**CIA MARKS**	**:**	**50**
COURSE TITLE	**:**	**DATA STRUCTURES AND ALGORITHMS PRACTICAL**	**SEE MARKS**	**:**	**50**
COURSE CODE	**:**	**BCA301P/BSC301P**	**TOTAL MARKS**	**:**	**100**
HOURS/WEEK	**:**	**4**	**CREDITS**	**:**	**1**

LAB LIST

1. Write a program to demonstrate linear search.
2. Write a program to demonstrate binary search.
3. Write a program to demonstrate selection sort.
4. Write a program to demonstrate insertion sort.
5. Write a program to demonstrate bubble sort.
6. Write a program to demonstrate merge sort.
7. Write a program to demonstrate quick sort.
8. Write a program to implement stack using array.
9. Write a program to implement queue using arrays
10. Write a program to implement a linked list and perform following operations.
    a. Insertion of a new node in the beginning of linked list
    b. Insertion of a new node at a given position of linked list
    c. Insertion of a new node at the end of linked list
    d. Deletion of a node from the beginning of linked list
    e. Deletion of a node from a given position of linked list
    f. Deletion of a node from the end of linked list
    g. Searching for a node in the linked list
    h. Display all the elements in the linked list
11. Write a program to evaluate postfix expressions.
12. Write a program to perform in order, preorder and post order traversal of a binary tree.
13. Write a program to implement Breadth First Search.
14. Write a program to implement Depth First Search.

1. Write a program to implement stack using array.

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

int stack[10], n, ele, x, i, top=-1, opt;

void push()
{
	if(top>=n-1)
		printf("Stack Overflow\n");
	else
	{
		printf("Enter element\n");
		scanf("%d",&ele);
		top=top+1;
		stack[top]=ele;
	}
}


void pop()
{
	if(top==-1)
		printf("Stack Underflow\n");
	else
	{
		x=stack[top];
		top=top-1;
		printf("Deleted element is %d\n", x);
	}
}
```

```
void display()
{
	if(top == -1)
		printf("Stack is empty\n");
	else
		printf("Stack elements are:\n");
		for(i=top;i>=0;i--)
		{
			printf("%d\n",stack[i]);
		}
}


void main ()
{
	printf("Enter the size of stack :\n");
	scanf("%d",&n);
	printf("*********Stack operations using array*********");
	while(opt!=4)
	{
		printf("Select any option\n");
		printf("\n1.Push\n2.Pop\n3.Display\n4.Exit");
		printf("\n Enter option number \n");
		scanf("%d",&opt);
		switch(opt)
		{
			case 1: push();
				break;
			case 2: pop();
				break;
			case 3: display();
				break;
			case 4: printf("Exit\n");
				exit(0);
			default: printf("Invalid choice \n");
		}
	}
}
```

OUTPUT

```
Enter the size of stack :
3
*********Stack operations using array*********
Select any option
1.Push
2.Pop
3.Display
4.Exit
 Enter option number: 1
Enter element
11

Select any option
1.Push
2.Pop
3.Display
4.Exit
 Enter option number: 1
Enter element
22

Select any option
1.Push
2.Pop
3.Display
4.Exit
 Enter option number: 1
Enter element
33

Select any option
1.Push
2.Pop
3.Display
4.Exit
 Enter option number: 1
Stack Overflow
```

Select any option
1.Push
2.Pop
3.Display
4.Exit
Enter option number: 3
Stack elements are:
33
22
11

Select any option
1.Push
2.Pop
3.Display
4.Exit
Enter option number: 2
Deleted element is 33

Select any option
1.Push
2.Pop
3.Display
4.Exit
Enter option number: 2
Deleted element is 22

Select any option
1.Push
2.Pop
3.Display
4.Exit
Enter option number: 2
Deleted element is 11

Select any option
1.Push
2.Pop
3.Display
4.Exit
Enter option number: 2
Stack Underflow

Select any option
1.Push
2.Pop
3.Display
4.Exit
Enter option number: 3
Stack is empty

Select any option
1.Push
2.Pop
3.Display
4.Exit
Enter option number: 4
Exit

2. Write a program to implement queue using arrays.

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

#define maxsize 2

int front = -1, rear = -1;
int queue[maxsize];

void insert()
{
        int item;
        if(rear == maxsize-1)
        {
                printf("\n OVERFLOW\n");
                return;
        }
        if(front == -1 && rear == -1)
        {
                printf("\n Enter the element\n");
                scanf("\n %d", &item);
                front = 0;
                rear = 0;
        }
        else
        {
                printf("\n Enter the element\n");
                scanf("\n %d", &item);
                rear = rear+1;
        }
        queue[rear] = item;
        printf("\n Value inserted\n");
}
```

```
void delete()
{
	int item;
	if (front == -1 || front > rear)
	{
		printf("\n UNDERFLOW\n");
		return;
	}
	else
	{
		item = queue[front];
		if(front == rear)
		{
			front = -1;
			rear = -1 ;
		}
		else
		{
			front = front + 1;
		}
		printf("\n value deleted is: %d\n",item);
	}
}


void display()
{
	int i;
	if(rear == -1)
	{
		printf("\n Empty queue\n");
	}
	else
	{
		printf("\n Queue elements are:\n");
		for(i=front;i<=rear;i++)
		{
			printf("%d\t",queue[i]);
		}
		printf("\n");
	}
```

```
int main ()
{
	int choice;
	while(choice != 4)
	{
		printf("\n**********Queue Operations Using Array**********\n");
		printf("\n1.Insert an element\n2.Delete an element\n3.Display the
			queue\n4.Exit\n");
		printf("\n Enter your choice : ");
		scanf("%d", &choice);
		switch(choice)
		{
			case 1: insert();
				break;
			case 2: delete();
				break;
			case 3: display();
				break;
			case 4: exit(0);
				break;
			default: printf("\n Enter valid choice\n");
		}
	}
}
```

OUTPUT

**********Queue Operations Using Array**********

1.Insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice : 1

Enter the element
11

Value inserted

**********Queue Operations Using Array**********

1.Insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice : 1

Enter the element
22

Value inserted

**********Queue Operations Using Array**********

1.Insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice : 1

OVERFLOW

**********Queue Operations Using Array**********

1.Insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice : 3

Queue elements are:
11 22

**********Queue Operations Using Array**********

1.Insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice : 2

value deleted is: 11

**********Queue Operations Using Array**********

1.Insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice : 2

value deleted is: 22

**********Queue Operations Using Array**********

1.Insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice : 2

UNDERFLOW

**********Queue Operations Using Array**********

1.Insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice : 3

Empty queue

**********Queue Operations Using Array**********

1.Insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice : 4

3. Write a program to evaluate postfix expressions.

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

#define MAX 100

int stack[MAX];
int top = -1;


void push(int x)
{
        if (top < MAX - 1)
        {
                stack[++top] = x;
        }
        else
        {
                printf("Stack overflow\n");
        }
}


int pop()
{
        if (top >= 0)
        {
                 return stack[top--];
        }
        else
        {
                printf("Stack underflow\n");
                return -1;
        }
}
```

```
int main()
{
	char exp[MAX];
	int i = 0, op1, op2;

	printf("Enter postfix expression: ");
	scanf("%s", exp);

	while (exp[i] != '\0')
	{
		if (isdigit(exp[i]))
		{
			push(exp[i] - '0');
		}
		else
		{
			op2 = pop();
			op1 = pop();
			switch (exp[i])
			{
			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;
			}
		}
		i++;
	}

	printf("After evaluation we get the value: %d\n", pop());
	return 0;
}
```

OUTPUT

Enter postfix expression: 52+
After evaluation we getthe value: 7

Enter postfix expression: 12*34*-
After evaluation we getthe value: -10

Enter postfix expression: 236*+1+
After evaluation we getthe value: 21

4. Write a program to implement a linked list and perform following operations.

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

struct node
{
        int info;
        struct node *next;
} *start;

struct node* create_node(int);
void insert_begin();
void insert_last();
void insert_pos();
void delete_begin();
void delete_last();
void delete_pos();
void search();
void display();

int main()
{
        int choice;
        start = NULL;

        do
        {
                printf("---------------------------------\n");
                printf("Operations on singly linked list\n");
                printf("---------------------------------\n");
                printf("1. Insert at first\n");
                printf("2. Insert at last\n");
                printf("3. Insert at position\n");
                printf("4. Delete at first\n");
                printf("5. Delete at Last\n");
                printf("6. Delete at position\n");
                printf("7. Search\n");
                printf("8. Display\n");
                printf("9. Exit\n");
```

```
printf("Enter your choice: ");
scanf("%d", &choice);

switch (choice)
{
        case 1:
                insert_begin();
                display();
                break;
        case 2:
                 insert_last();
                 display();
                 break;
        case 3:
                insert_pos();
                display();
                break;
        case 4:
                delete_begin();
                display();
                break;
        case 5:
                delete_last();
                display();
                break;
        case 6:
                delete_pos();
                display();
                break;
        case 7:
                search();
                display();
                break;
        case 8:
                display();
                break;
        case 9:
                exit(0);
                break;
        default:printf("Wrong choice...???\n");
                break;
}
```

```
        } while (choice != 9);

        return 0;
}

struct node* create_node(int value)
{
        struct node *temp;
        temp = (struct node*)malloc(sizeof(struct node));
        if (temp == NULL)
        {
                printf("Memory not allocated\n");
                return NULL;
        }
        else
        {
                temp->info = value;
                temp->next = NULL;
                return temp;
        }
}

void insert_begin()
{
        int value;
        printf("Enter the value to be inserted: ");
        scanf("%d", &value);
        struct node *temp, *s;
        temp = create_node(value);
        if (start == NULL)
        {
                start = temp;
                start->next = NULL;
                printf("%d is inserted at first in the empty list\n", temp->info);
        }
        else
        {
                s = start;
                start = temp;
                start->next = s;
                printf("%d is inserted at first\n", temp->info);
        }
```

```
}

void insert_last()
{
	int value;
	printf("Enter the value to be inserted: ");
	scanf("%d", &value);
	struct node *temp, *s;
	temp = create_node(value);
	if (start == NULL)
	{
		start = temp;
		start->next = NULL;
		printf("%d is inserted at last in the empty list\n", temp->info);
	}
	else
	{
		s = start;
		while (s->next != NULL)
		{
			s = s->next;
		}
		temp->next = NULL;
		s->next = temp;
		printf("%d is inserted at last\n", temp->info);
	}
}

void insert_pos()
{
	int value, pos, counter = 0, loc = 1;
	struct node *temp, *s, *ptr;
	s = start;
	while (s != NULL)
	{
		s = s->next;
		counter++;
	}
	if (counter == 0)
	{
		printf("List is empty\n");
	}
```

```
else
{
        printf("Enter the position from %d to %d: ", loc, counter + 1);
        scanf("%d", &pos);
        s = start;
        if (pos == 1)
        {
                printf("Enter the value to be inserted: ");
                scanf("%d", &value);
                temp = create_node(value);
                start = temp;
                start->next = s;
                printf("%d is inserted at first\n", temp->info);
        }
        else if (pos > 1 && pos <= counter)
        {
                printf("Enter the value to be inserted: ");
                scanf("%d", &value);
                temp = create_node(value);
                for (int i = 1; i < pos; i++)
                {
                        ptr = s;
                        s = s->next;
                }
                ptr->next = temp;
                temp->next = s;
                printf("%d is inserted at position %d\n", temp->info, pos);
        }
        else if (pos == counter + 1)
        {
                printf("Enter the value to be inserted: ");
                scanf("%d", &value);
                temp = create_node(value);
                while (s->next != NULL)
                {
                        s = s->next;
                }
                temp->next = NULL;
                s->next = temp;
                printf("%d is inserted at last\n", temp->info);
        }
        else
```

```
                {
                        printf("Position out of range...!!!\n");
                }
        }
}

void delete_begin()
{
        if (start == NULL)
        {
                printf("List is empty\n");
        }
        else
        {
                struct node *s;
                s = start;
                start = s->next;
                printf("%d deleted from first\n", s->info);
                free(s);
        }
}

void delete_last()
{
        int i, counter = 0;
        struct node *s, *ptr;
        if (start == NULL)
        {
                printf("List is empty\n");
        }
        else
        {
                s = start;
                while (s != NULL)
                {
                        s = s->next;
                        counter++;
                }
                s = start;
                if (counter == 1)
                {
                        start = s->next;
```

```
                printf("%d deleted from last\n", s->info);
                free(s);
            }
            else
            {
                for (i = 1; i < counter; i++)
                {
                    ptr = s;
                    s = s->next;
                }
                ptr->next = s->next;
                printf("%d deleted from last\n", s->info);
                free(s);
            }
        }
}

void delete_pos()
{
    int pos, i, counter = 0, loc = 1;
    struct node *s, *ptr;
    s = start;
    while (s != NULL)
    {
        s = s->next;
        counter++;
    }
    if (counter == 0)
    {
        printf("List is empty\n");
    }
    else
    {
        printf("Enter the position from %d to %d: ", loc, counter);
        scanf("%d", &pos);
        s = start;
        if (pos == 1)
        {
            start = s->next;
            printf("%d deleted from first\n", s->info);
            free(s);
        }
```

```
            else if (pos > 1 && pos <= counter)
            {
                for (i = 1; i < pos; i++)
                {
                    ptr = s;
                    s = s->next;
                }
                ptr->next = s->next;
                if (pos == counter)
                {
                    printf("%d deleted from last\n", s->info);
                     free(s);
                }
                else
                {
                    printf("%d deleted from position %d\n", s->info, pos);
                    free(s);
                }
            }
            else
            {
                printf("Position out of range...!!!\n");
            }
        }
}

void search()
{
        int value, loc = 0, pos = 0, counter = 0;
        struct node *s;
        s = start;
        while (s != NULL)
        {
            s = s->next;
            counter++;
        }
        if (start == NULL)
        {
            printf("List is empty\n");
        }
        else
        {
```

```
                printf("Enter the value to be searched: ");
                scanf("%d", &value);
                s = start;
                while (s != NULL)
                {
                        pos++;
                        if (s->info == value)
                        {
                                loc++;
                                if (loc == 1)
                                {
                                        printf("Element %d is found at position %d", value, pos);
                                }
                                else if (loc <= counter)
                                {
                                        printf(" , %d", pos);
                                }
                        }
                        s = s->next;
                }
                printf("\n");
                if (loc == 0)
                {
                        printf("Element %d not found in the list\n", value);
                }
        }
}

void display()
{
        struct node *temp;
        if (start == NULL)
        {
                printf("Linked list is empty...!!!\n");
        }
        else
        {
                printf("Linked list contains: ");
                temp = start;
                while (temp != NULL)
                {
                        printf("%d ", temp->info);
```

```
                temp = temp->next;
            }
            printf("\n");
        }
}
```

OUTPUT

```
---------------------------------
Operations on singly linked list
---------------------------------
1. Insert at first
2. Insert at last
3. Insert at position
4. Delete at first
5. Delete at Last
6. Delete at position
7. Search
8. Display
9. Exit
Enter your choice: 1
Enter the value to be inserted: 10
10 is inserted at first in the empty list
Linked list contains: 10
---------------------------------
Operations on singly linked list
---------------------------------
1. Insert at first
2. Insert at last
3. Insert at position
4. Delete at first
5. Delete at Last
6. Delete at position
7. Search
8. Display
9. Exit
Enter your choice: 1
Enter the value to be inserted: 20
20 is inserted at first
Linked list contains: 20 10
```

```
---------------------------------
Operations on singly linked list
---------------------------------
1. Insert at first
2. Insert at last
3. Insert at position
4. Delete at first
5. Delete at Last
6. Delete at position
7. Search
8. Display
9. Exit
Enter your choice: 2
Enter the value to be inserted: 30
30 is inserted at last
Linked list contains: 20 10 30
---------------------------------
Operations on singly linked list
---------------------------------
1. Insert at first
2. Insert at last
3. Insert at position
4. Delete at first
5. Delete at Last
6. Delete at position
7. Search
8. Display
9. Exit
Enter your choice: 3
Enter the position from 1 to 4: 2
Enter the value to be inserted: 40
40 is inserted at position 2
Linked list contains: 20 40 10 30
---------------------------------
Operations on singly linked list
---------------------------------
1. Insert at first
2. Insert at last
3. Insert at position
4. Delete at first
5. Delete at Last
6. Delete at position
```

```
7. Search
8. Display
9. Exit
Enter your choice: 8
Linked list contains: 20 40 10 30
---------------------------------
Operations on singly linked list
---------------------------------
1. Insert at first
2. Insert at last
3. Insert at position
4. Delete at first
5. Delete at Last
6. Delete at position
7. Search
8. Display
9. Exit
Enter your choice: 7
Enter the value to be searched: 50

Element 50 not found in the list
Linked list contains: 20 40 10 30
---------------------------------
Operations on singly linked list
---------------------------------
1. Insert at first
2. Insert at last
3. Insert at position
4. Delete at first
5. Delete at Last
6. Delete at position
7. Search
8. Display
9. Exit
Enter your choice: 7
Enter the value to be searched: 30
Element 30 is found at position 4
Linked list contains: 20 40 10 30
---------------------------------
Operations on singly linked list
---------------------------------
1. Insert at first
```

```
2. Insert at last
3. Insert at position
4. Delete at first
5. Delete at Last
6. Delete at position
7. Search
8. Display
9. Exit
Enter your choice: 4
20 deleted from first
Linked list contains: 40 10 30
---------------------------------
Operations on singly linked list
---------------------------------
1. Insert at first
2. Insert at last
3. Insert at position
4. Delete at first
5. Delete at Last
6. Delete at position
7. Search
8. Display
9. Exit
Enter your choice: 6
Enter the position from 1 to 3: 2
10 deleted from position 2
Linked list contains: 40 30
---------------------------------
Operations on singly linked list
---------------------------------
1. Insert at first
2. Insert at last
3. Insert at position
4. Delete at first
5. Delete at Last
6. Delete at position
7. Search
8. Display
9. Exit
Enter your choice: 5
30 deleted from last
Linked list contains: 40
```

```
---------------------------------
Operations on singly linked list
---------------------------------
1. Insert at first
2. Insert at last
3. Insert at position
4. Delete at first
5. Delete at Last
6. Delete at position
7. Search
8. Display
9. Exit
Enter your choice: 5
40 deleted from last
Linked list is empty...!!!
---------------------------------
Operations on singly linked list
---------------------------------
1. Insert at first
2. Insert at last
3. Insert at position
4. Delete at first
5. Delete at Last
6. Delete at position
7. Search
8. Display
9. Exit
Enter your choice: 5
List is empty
Linked list is empty...!!!
---------------------------------
Operations on singly linked list
---------------------------------
1. Insert at first
2. Insert at last
3. Insert at position
4. Delete at first
5. Delete at Last
6. Delete at position
7. Search
8. Display
9. Exit
```

```
Enter your choice: 8
Linked list is empty...!!!
---------------------------------
Operations on singly linked list
---------------------------------
1. Insert at first
2. Insert at last
3. Insert at position
4. Delete at first
5. Delete at Last
6. Delete at position
7. Search
8. Display
9. Exit
Enter your choice: 9
```

5. Write a program to demonstrate linear search.

```
#include<stdio.h>
int main()
{
        int a[20],i,x,n;
        printf("How many elements?");
        scanf("%d",&n);
        printf("Enter array elements:\n");
        for(i=0;i<n;++i)
                scanf("%d",&a[i]);
        printf("\n Enter element to search:");
        scanf("%d",&x);
        for(i=0;i<n;++i)
                if(a[i]==x)
                        break;
        if(i<n)
                printf("Element found at index %d",i);
        else
                printf("Element not found");
        return 0;
}
```

OUTPUT

```
How many elements?5
Enter array elements:
10 20 30 40 50
Enter element to search:50
Element found at index 4
```

6. Write a program to demonstrate binary search.

```
#include <stdio.h>
int main()
{
	int i, low, high, mid, n, key, array[100];
	printf("Enter number of elements \n");
	scanf("%d",&n);
	printf("Enter %d integers \n",n);
	for(i = 0; i< n; i++)
		scanf("%d",&array[i]);
	printf("Enter value to find \n");
	scanf("%d", &key);
	low = 0;
	high = n - 1;
	mid = (low+high)/2;
	while (low <= high)
	 {
	I	f(array[mid] < key)
			low = mid + 1;
		else if (array[mid] == key)
		{
			printf("%d found at position %d\n", key, mid+1);
			break;
		}
		else
			high = mid - 1;
			mid = (low + high)/2;
	}
	if(low > high)
		printf("%d is not found in array\n", key);
	return 0;
}
```

OUTPUT

```
Enter number of elements
5
Enter 5 integers
10 20 30 40 50
Enter value to find
40
40 found at position 4
```

7. Write a program to demonstrate selection sort.

```
#include <stdio.h>
int main()
{
        int arr[10]={6,12,0,18,11,99,55,45,34,2};
        int n=10;
        inti, j, position, swap;
        printf("Array elements before sorting");
        for (i = 0; i< n; i++)
                printf("%d\t", arr[i]);
        for (i = 0; i< (n - 1); i++)
        {
                position = i;
                for (j = i + 1; j < n; j++)
                {
                        if (arr[position] >arr[j])
                        position = j;
                }
                if (position != i)
                {
                        swap = arr[i];
                        arr[i] = arr[position];
                        arr[position] = swap;
                }
        }
        printf("Array elements after sorting");
        for (i = 0; i< n; i++)
                printf("%d\t", arr[i]);
        return 0;
}
```

OUTPUT

```
Array elements before sorting
6       12      0       18      11      99      55      45      34      2
Array elements after sorting
0       2       6       11      12      18      34      45      55      99
```

8. Write a program to demonstrate insertion sort.

```
#include <stdio.h>
int main(void)
{
        int n, i, j, temp;
        intarr[15];
        printf("Enter number of elements\n");
        scanf("%d", &n);
        printf("Enter %d integers\n", n);
        for (i = 0; i< n; i++)
                scanf("%d", &arr[i]);
        for (i = 1; i< n; i++)
        {
                j = i;
                while (j > 0 &&arr[j - 1] >arr[j])
        {
                        temp = arr[j];
                        arr[j] = arr[j - 1];
                        arr[j - 1] = temp;
                        j--;
                }
        }
        printf("Sorted list in ascending order:\n");
        for (i = 0; i< n; i++)
                printf("%d\n", arr[i]);
       return 0;
}
```

OUTPUT

```
Enter number of elements
5
Enter 5 integers
50 40 30 20 10
Sorted list in ascending order:
10
20
30
40
50
```

9. Write a program to demonstrate bubble sort.

```
#include <stdio.h>

int main()
{
int array[100], n, i, j, swap;
printf("Enter number of elements\n");
scanf("%d", &n);
printf("Enter %d integers\n", n);
for (i= 0; i< n; i++)
        scanf("%d", &array[i]);
for (i= 0 ;i< n - 1; i++)
{
        for (j = 0 ; j< n - i - 1; j++)
        {
                if (array[j] > array[j+1]) /* For decreasing order use '<' instead of '>' */
                {
                        swap      = array[j];
                        array[j]   = array[j+1];
                        array[j+1] = swap;
                }
        }
 }
printf("Sorted list in ascending order:\n");
 for (i = 0; i< n; i++)
        printf("%d\n", array[i]);
 return 0;
}
```

OUTPUT

```
Enter number of elements
5
Enter 5 integers
20 15 10 5 30
Sorted list in ascending order:
5
10
15
20
30
```

10. Write a program to demonstrate merge sort.

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

void mergesort(int a[], int p, int r)
{
	if(p<r)
	{
		int q=(p+r)/2;
		mergesort(a,p,q);
		mergesort(a,q+1,r);
		merge (a,p,q,r);
	}
}

void merge(int a[], int p, int q, int r)
{
	int c[10];
	int i=p;
	int j=q+1;
	int k=p;
	while((i<=q)&&(j<=r))
	{
		if(a[i]<a[j])
		{
			c[k]=a[i];
			i=i+1;
			k=k+1;
		}
		else
		{
			c[k]=a[j];
			j=j+1;
			k=k+1;
		}
	}
	while(i<=q)
	{
		c[k]=a[i];
		i=i+1;
		k=k+1;
```

```
        }
        while(j<=r)
        {
                c[k]=a[j];
                j=j+1;
                k=k+1;
        }
        int l=p;
        while(l<=r)
        {
                a[l]=c[l];
                l=l+1;
        }
}

int main()
{
        int a[10],I,p,r,n;
        printf("Enter the size of array\n");
        scanf("%d",&n);
        p=0;
        r=n-1;
        printf("Enter the array elements\n");
        for(i=0;i<n;i++)
                scanf("%d",&a[i]);
        mergesort(a,p,r);
        printf("Array elements after sorting\n");
        for(i=0;i<n;i++)
                printf("%d",a[i]);
        return 0;
}
```

OUTPUT

```
Enter the size of array
5
Enter array elements
2       7       5       1       9
Array elements after sorting
1
2
5
7
9
```

11. Write a program to demonstrate quick sort.

```
#include<stdio.h>

void quicksort(int number[25],int first,int last)
{
	int i, j, pivot, temp;
	if(first<last)
	{
		pivot=first;
		i=first;
		j=last;
		while(i<j)
		{
			while(number[i]<=number[pivot]&&i<last)
				i++;
			while(number[j]>number[pivot])
				j--;
			if(i<j)
			{
				temp=number[i];
				number[i]=number[j];
				number[j]=temp;
			}
		}
		temp=number[pivot];
		number[pivot]=number[j];
		number[j]=temp;
		quicksort(number,first,j-1);
		quicksort(number,j+1,last);
	}
}
```

```
int main()
{
        inti, count, number[25];
        printf("Enter array size (Max. - 25): ");
        scanf("%d",&count);
        printf("Enter %d elements: ", count);
        for(i=0;i<count;i++)
                scanf("%d",&number[i]);
        quicksort(number,0,count-1);
        printf("The Sorted Order is: ");
        for(i=0;i<count;i++)
                printf(" %d",number[i]);
        return 0;
}
```

OUTPUT

```
Enter array size (Max. - 25): 5
Enter 5 elements: 20 40 10 5 15
The Sorted Order is:  5 10 15 20 40
```

12. Write a program to perform in order, preorder and post order traversal of a binary tree.

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

// Structure of a node of tree
struct node
{
        int data;
        struct node* left;
        struct node* right;
};

// Create a New Node
struct node* createNode(int data)
{
// Allocate memory equivalent to node structure and hold address in node pointer
        struct node* newNode = malloc(sizeof(struct node));
        newNode -> data = data;
        newNode -> left = NULL;
        newNode -> right = NULL;
        return newNode;
}

// Insert a new node to left of the given node
struct node* insertLeft(struct node* root, int data)
{
        root -> left = createNode(data);
        return root;
}

// Insert a new node to right of the given node
struct node* insertRight(struct node* root, int data)
{
        root -> right = createNode(data);
        return root;
}
```

```
// Inorder traversal
void inorder(struct node* root)
{
	if (root == NULL)
		return;
	inorder(root -> left);
	printf("%d ", root -> data);
	inorder(root -> right);
}

// Preorder traversal
void preorder(struct node* root)
{
	if (root == NULL)
		return;
	printf("%d ", root -> data);
	preorder(root -> left);
	preorder(root -> right);
}

// Postorder traversal
void postorder(struct node* root)
{
	if (root == NULL)
		return;
	postorder(root -> left);
	postorder(root -> right);
	printf("%d ", root -> data);
}

int main()
{
	struct node* root = createNode(4);
	insertLeft(root, 21);
	insertRight(root, 13);
	insertLeft(root -> left, 34);
	insertRight(root -> left, 0);
	insertLeft(root -> right, 18);
	insertRight(root -> right, 19);
	printf("Inorder traversal of the Tree is : ");
	inorder(root);
```

```
        printf("\n");
        printf("Preorder traversal of the Tree is : ");
        preorder(root);
        printf("\n");
        printf("Postorder traversal of the Tree is : ");
        postorder(root);
        return 0;
}
```

OUTPUT

```
Inorder traversal of the Tree is : 34 21 0 4 18 13 19
Preorder traversal of the Tree is : 4 21 34 0 13 18 19
Postorder traversal of the Tree is : 34 0 21 18 19 13 4
```

13. Write a program to implement Breadth First Search.

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

#define MAX 100

int queue[MAX], front = -1, rear = -1;
int visited[MAX] = {0};
int adjMatrix[MAX][MAX];
int numVertices;

void enqueue(int vertex)
{
	if (rear == MAX - 1)
	{
		printf("Queue overflow\n");
		return;
	}
	if (front == -1)
		front = 0;
	queue[++rear] = vertex;
}

int dequeue()
{
	if (front == -1 || front > rear)
	{
		printf("Queue underflow\n");
		return -1;
	}
	return queue[front++];
}

int isQueueEmpty()
{
	return front == -1 || front > rear;
}
```

```
void BFS(int startVertex)
{
	enqueue(startVertex);
	visited[startVertex] = 1;

	while (!isQueueEmpty())
	{
		int currentVertex = dequeue();
		printf("%d ", currentVertex);

		for (inti = 0; i<numVertices; i++)
		{
			if (adjMatrix[currentVertex][i] == 1 && !visited[i])
			{
			enqueue(i);
			visited[i] = 1;
			}
		}
	}
}

int main()
{
	int startVertex;
	printf("Enter the number of vertices in the graph: ");
	scanf("%d", &numVertices);
	printf("Enter the adjacency matrix of the graph:\n");
	for (int i = 0; i<numVertices; i++)
	{
		for (int j = 0; j <numVertices; j++)
		{
			scanf("%d", &adjMatrix[i][j]);
		}
	}
	printf("Enter the starting vertex for BFS: ");
	scanf("%d", &startVertex);
	printf("BFS traversal starting from vertex %d: ", startVertex);
	BFS(startVertex);
	printf("\n");
	return 0;
}
```

OUTPUT

```
Enter the number of vertices in the graph: 4
Enter the adjacency matrix of the graph:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
Enter the starting vertex for BFS: 0
BFS traversal starting from vertex 0: 0 1 2 3
```

OR

```
#include<stdio.h>

int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;
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++]);
		}
}

int main()
{
	int v;
	printf("Enter the number of vertices: ");
	scanf("%d",&n);
	for(i=1; i<= n; i++)
	{
		q[i] = 0;
		visited[i] = 0;
	}
	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]);
        }
        printf("Enter the starting vertex: ");
        scanf("%d", &v);
        bfs(v);
        printf("\nThe node which are reachable are:");
        for(i=1; i<= n; i++)
        {
                if(visited[i])
                        printf(" %d", i);
                else
                {
                        printf("\nBFS is not possible. All nodes are not reachable!");
                        break;
                }
        }
}
```

OUTPUT

```
Enter the number of vertices: 4
Enter graph data in matrix form:
1 1 1 1
0 1 0 0
0 0 1 0
0 0 0 1
Enter the starting vertex: 1
The node which are reachable are: 1 2 3 4
```

14. Write a program to implement Depth First Search.

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

#define MAX 100

int stack[MAX];
int top = -1;
int visited[MAX] = {0};
int adjMatrix[MAX][MAX];
int numVertices;

void push(int vertex) {
   if (top == MAX - 1) {
printf("Stack overflow\n");
      return;
   }
   stack[++top] = vertex;
}

int pop() {
   if (top == -1) {
printf("Stack underflow\n");
      return -1;
   }
   return stack[top--];
}

int isStackEmpty() {
   return top == -1;
}

void DFS(int startVertex) {
   push(startVertex);
   visited[startVertex] = 1;

   while (!isStackEmpty()) {
intcurrentVertex = pop();
printf("%d ", currentVertex);

      for (inti = 0; i<numVertices; i++) {
```

```
            if (adjMatrix[currentVertex][i] == 1 && !visited[i]) {
               push(i);
               visited[i] = 1;
            }
         }
      }
   }

int main() {
int startVertex;

printf("Enter the number of vertices in the graph: ");
scanf("%d", &numVertices);

printf("Enter the adjacency matrix of the graph:\n");
   for (inti = 0; i<numVertices; i++) {
      for (int j = 0; j <numVertices; j++) {
scanf("%d", &adjMatrix[i][j]);
      }
   }

printf("Enter the starting vertex for DFS: ");
scanf("%d", &startVertex);

printf("DFS traversal starting from vertex %d: ", startVertex);
   DFS(startVertex);
printf("\n");

   return 0;
}
```

OUTPUT

```
Enter the number of vertices in the graph: 4
Enter the adjacency matrix of the graph:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
Enter the starting vertex for DFS: 0
DFS traversal starting from vertex 0: 0 2 3 1
```

Additional programs (if time permits):

15. Write a C program to implement stack using linked list.
16. Write a C program to implement queue using linked list.
17. Write a C program to perform infix to postfix conversion.
18. Write a C program to perform the following operations on singly linked list:
    a. Update a node in given position
    b. Sort the elements in ascending order
    c. Reverse the elements
    d. Display the elements
18. Write a C program to perform insertion and deletion operations in binary tree.