ATME
College of Engineering

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama",Belagavi-590018

ATME COLLEGE OF ENGINEERING

13th Kilometer, Mysore-Kanakapura-Bangalore Road Mysore-570028

Department of Computer Science and Engineering
(Artificial Intelligence & Machine Learning)

Assignment
on

"Data Structures and Applications"
Course Code: BCS304
Branch: CSE-AI&ML
Semester: 3rd

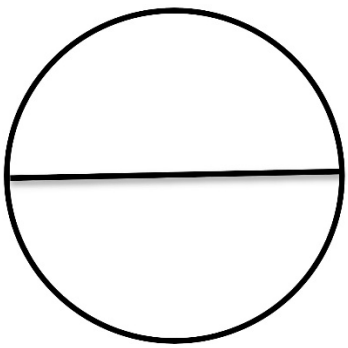
Submitted by:
USN: 4AD24CI039
NAME:Pragna P

Submitted To:
Dr Anil Kumar C J
Head Of the Department
CSE-AIML

List of Programs

Program No.	Program Description	PAGE NO
1	Initialization of malloc () function into 0	4
2	Printing the address of the pointer	4
3	Selection sort with dynamic memory allocation	5
4	Selection sort without dynamic memory allocation	6
5	Declaration of two-dimensional array using dynamic memory allocation	7
6	Pattern matching (general way)	8-9
7	Self-Referential structure	10
8	Pre increment and Post increment	11
9	Post decrement and Pre decrement	12
10	Regular Queue	13-14
11	Operations of linked lists (traversal, insertion, deletion)	14-16
12	Linked list using Stacks	16-18
13	Linked list using Queues	18-20
14	Time taken for arrays and linked lists during insertion and deletion	21-22
15	Sparse Matrix	23-24
16	Polynomial Representation	24-26
17	How to create a tree	26-27

18	Construct a tree using arrays	27-28
19	Construct a binary tree using queues	28-31
20	Insertion of nodes	31-34
21	Deletion of nodes	34-38
22	Depth search tree	38-39
23	Breadth search tree	39-41
24	Level order	41-43
25	DFS and BFS using adjacency list (use stacks and queues in program)	44-46
26	Calloc and Malloc functions checking if junk/zero is initialized to them	47
27	Circular linked lists basic operations	48-50
28	How to implement comparison of 2 strings using built in function	50-51
29	In linked list insertion in the middle and deletion in the middle	51-53
30	Binary tree traversal	53-55
31	Circular Queue using Array(Modulo division , Queue full , Queue empty)	55-57
32	Sparse Matrix representation using Linked List	58-59



PROGRAM 1:
Initialization of malloc () function into 0

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int m = (int)malloc(sizeof(int));
   if (m) *m = 0; // Manual initialization
   printf("malloc value: %d\n", *m);
    free(m);
   fflush(stdout);
   system("getmac");
   return 0;
}
```

OUTPUT:

```
malloc value: 0

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 3.280 s
Press any key to continue.
```

PROGRAM 2:
Printing the address of the pointer

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int x = 10;
   int *ptr = &x;
   printf("Address of x: %p\n", (void*)&x);
    printf("Address of ptr: %p\n", (void*)&ptr);
   fflush(stdout);
    system("getmac");
    return 0;
}
```

Output:

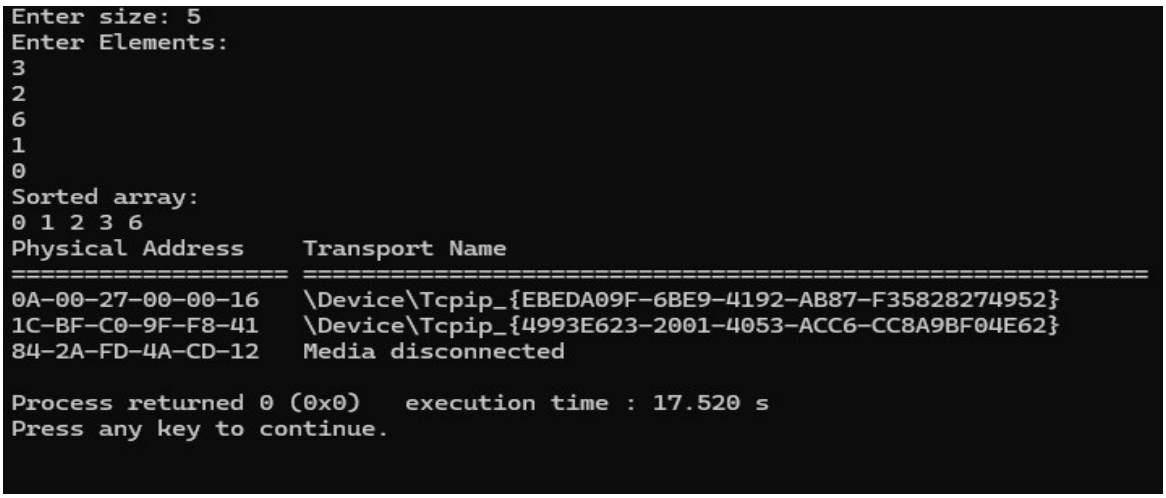
```
Address of x: 0061FF1C
Address of ptr: 0061FF18

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 2.229 s
Press any key to continue.
```

PROGRAM 3:

Selection sort with dynamic memory allocation

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int n, temp;
   printf("Enter size: ");
   scanf("%d", &n);
   printf("Enter Elements: ");
   int a = (int)malloc(n * sizeof(int));
   for(int i=0; i<n; i++)
      scanf("%d", &a[i]);
   for(int i=0; i<n-1; i++) {
      for(int j=i+1; j<n; j++) {
         if(a[i] > a[j]) {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
         }
      }
   }
   printf("Sorted array:\n");
   for(int i=0; i<n; i++)
      printf("%d ", a[i]);
free(a);

   fflush(stdout);
   system("getmac");
   return 0;
}
```

Output:



PROGRAM 4:
Selection sort without dynamic memory allocation

```
include <stdio.h>
#include <stdlib.h>
void selectionSort(int a[], int n)
{
   int min, temp;
   for(int i = 0; i < n - 1; i++)
   {
      min = i;
      for(int j = i + 1; j < n; j++)
      {
         if(a[j] < a[min])
            min = j;
      }
      temp = a[i];
      a[i] = a[min];
      a[min] = temp;
   }
}
int main()
{
   int a[20];
   int n;
   printf("Enter size: ");
   scanf("%d", &n);
   printf("Enter elements: ",&a);
   for(int i = 0; i < n; i++)
      scanf("%d", &a[i]);
   selectionSort(a, n);
   printf("Sorted array:\n");
   for(int i = 0; i < n; i++)
      printf("%d ", a[i]);

   printf("\nProgram Output: Success\n");
   fflush(stdout);         // flush output buffer
   system("getmac");       // works only on Windows

   return 0;
}
```

Output:

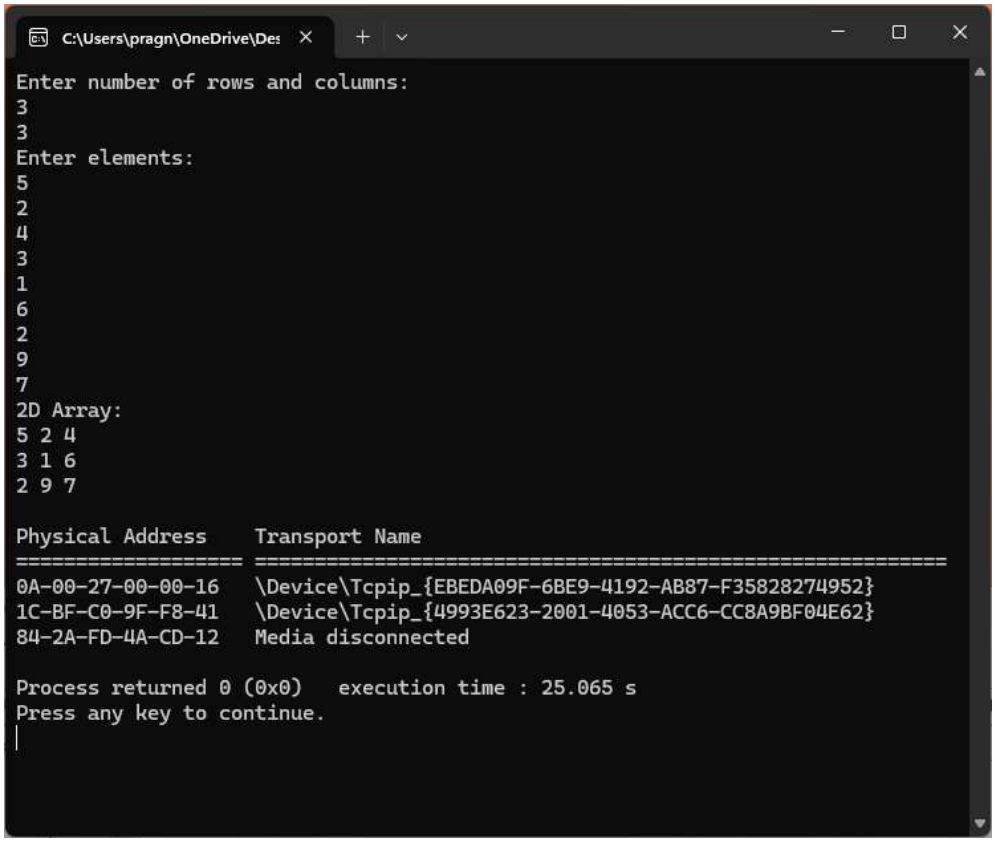
```
Enter size: 5
Enter elements:
6
2
4
3
1
Sorted array:
1 2 3 4 6
Program Output: Success

Physical Address    Transport Name
=================== ==========================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 13.164 s
Press any key to continue.
```

PROGRAM 5:

Declaration of two-dimensional array using dynamic memory allocation

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int rows, cols;
   printf("Enter number of rows and columns: ");
   scanf("%d %d", &rows, &cols);
   /* Dynamic memory allocation for 2D array */
   int *a = (int)malloc(rows * sizeof(int));
   for(int i = 0; i < rows; i++)
      a[i] = (int*)malloc(cols * sizeof(int));
   printf("Enter elements:\n");
   for(int i = 0; i < rows; i++)
      for(int j = 0; j < cols; j++)
         scanf("%d", &a[i][j]);
   printf("2D Array:\n");
   for(int i = 0; i < rows; i++) {
      for(int j = 0; j < cols; j++)
         printf("%d ", a[i][j]);
      printf("\n");
   }
   for(int i = 0; i < rows; i++)
      free(a[i]);
   free(a);
   fflush(stdout);
   system("getmac");
   return 0;
}
```

Output:

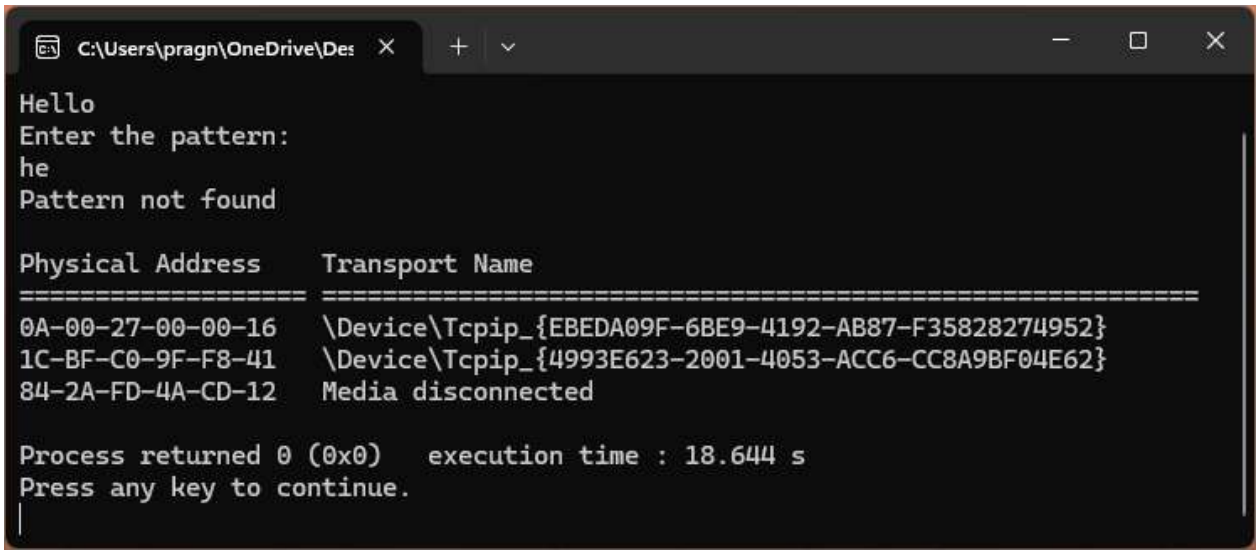


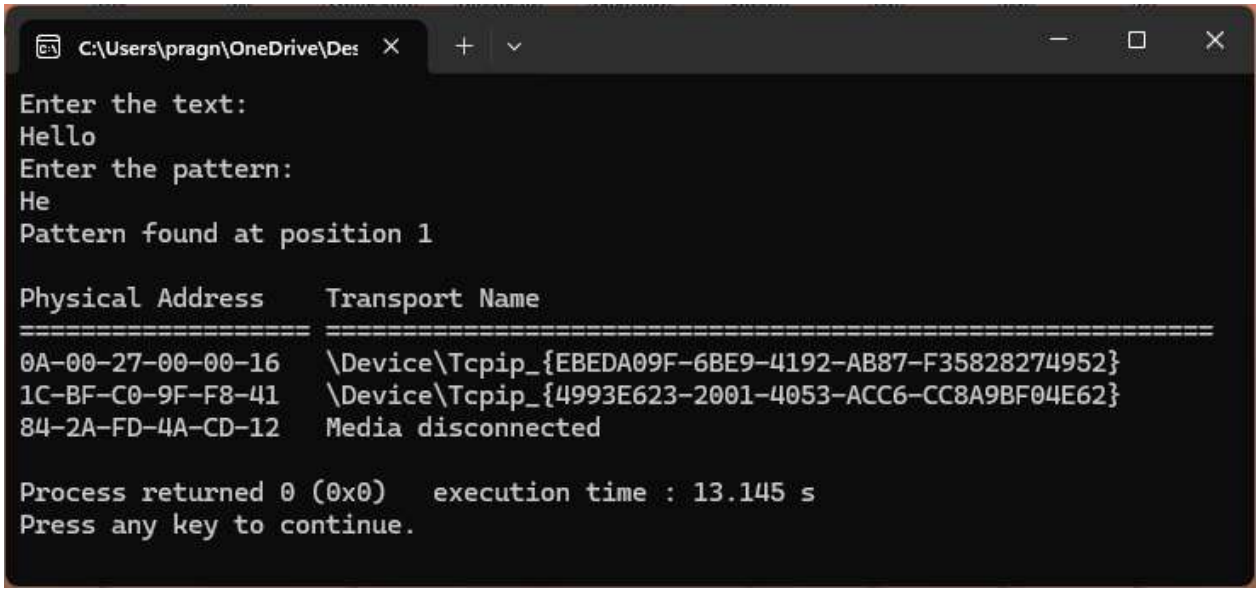
PROGRAM 6:

Pattern matching (general way)

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

int main()
{
   char text[100], pattern[50];
   int i, j, flag;

   printf("Enter the text: ");
   scanf("%s", text);

   printf("Enter the pattern: ");
   scanf("%s", pattern);
   int n = strlen(text);
   int m = strlen(pattern);

   for(i = 0; i <= n - m; i++)
   {
      flag = 1;
      for(j = 0; j < m; j++)
      {
         if(text[i + j] != pattern[j])
         {
            flag = 0;
            break;
         }
      }
      if(flag)
      {
         printf("Pattern found at position %d\n", i + 1);
         break;
      }
   }

   if(!flag)
      printf("Pattern not found\n");

   fflush(stdout);         // flush output buffer
   system("getmac");
   return 0;
}
```

Output:


C:\Users\pragn\OneDrive\Des
Hello
Enter the pattern:
he
Pattern not found
Physical Address Transport Name
0A-00-27-00-00-16 \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41 \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12 Media disconnected
Process returned 0 (0x0) execution time : 18.644 s
Press any key to continue.

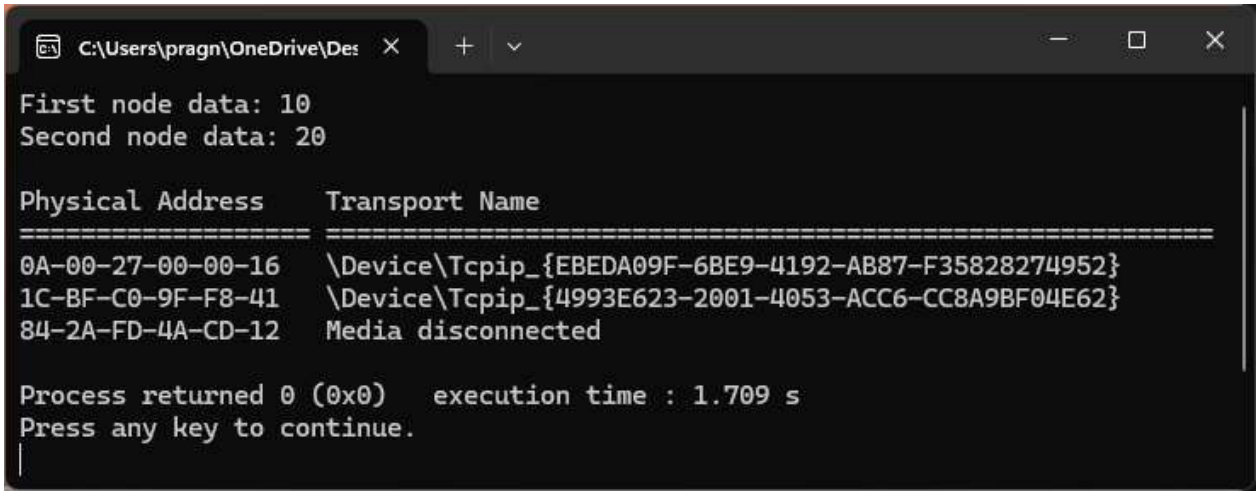

C:\Users\pragn\OneDrive\Des
Enter the text:
Hello
Enter the pattern:
He
Pattern found at position 1
Physical Address Transport Name
0A-00-27-00-00-16 \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41 \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12 Media disconnected
Process returned 0 (0x0) execution time : 13.145 s
Press any key to continue.

PROGRAM 7:

Self referential structure

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   struct node *next;   // self-referential pointer
};
int main()
{
   struct node n1, n2;

   n1.data = 10;
   n2.data = 20;

   n1.next = &n2;
   n2.next = NULL;

   printf("First node data: %d\n", n1.data);
   printf("Second node data: %d\n", n1.next->data);

   fflush(stdout);         // flush output buffer
   system("getmac");       // works only on Windows

   return 0;
}
```

Output:



PROGRAM 8:

Pre increment and Post increment

```
#include <stdio.h>
int main()
{
   int a;
   printf("Enter element: ");
   scanf("%d",&a);
   printf("Initial Value (A): %d",a);
   printf("\nPre Increment: %d", ++a);
   printf("\nPost Increment: %d", a++);
   printf("\n");
   fflush(stdout);
    system("getmac");
   return 0;
}
```

Output :

```
Enter element: 5
Initial Value (A): 5
Pre Increment: 6
Post Increment: 6

Physical Address    Transport Name
=================== ==========================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 6.388 s
Press any key to continue.
```

PROGRAM 9:

Post decrement and Pre decrement

```
#include <stdio.h>
int main()
{
   int a;
   printf("Enter element: ");
   scanf("%d",&a);
   printf("Initial Value (A): %d",a);
   printf("\nPre Decrement: %d", --a);
   printf("\nPost Decrement: %d", a--);
   printf("\n");
   fflush(stdout);         // flush output buffer
    system("getmac");
    return 0;
}
```

Output:

```
Enter element: 5
Initial Value (A): 5
Pre Decrement: 4
Post Decrement: 4

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 4.482 s
Press any key to continue.
```

PROGRAM 10:

Regular Queue

```
#include <stdio.h>
#include <stdlib.h>
int queue[100];
int front = -1, rear = -1, size;
void enqueue(int x)
{
   if (rear == size - 1)
   {
      printf("Queue is Full\n");
      return;
   }

   if (front == -1)
      front = 0;

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

   printf("Deleted element: %d\n", queue[front++]);
}

void display()
{
   if (front == -1 || front > rear)
   {
      printf("Queue is Empty\n");
      return;
   }

   printf("Queue elements: ");
   for (int i = front; i <= rear; i++)
      printf("%d ", queue[i]);
   printf("\n");
}

int main()
{
   int n, value;

   printf("Enter number of elements to insert: ");
   scanf("%d", &n);

   printf("Enter elements:\n");
   for (int i = 0; i < n; i++)
```

```
    {
        scanf("%d", &value);
        enqueue(value);
    }
    display();
    dequeue();
    display();
    fflush(stdout);         // flush output buffer
    system("getmac");       // works only on Windows
    return 0;
}
```

Output:

```
Enter queue size: 5
Enter number of elements to insert: 3
Enter elements:
1 3 2
Queue elements: 1 3 2
Deleted element: 1
Queue elements: 3 2

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 9.571 s
Press any key to continue.
```

PROGRAM 11:

Operations of linked lists (traversal, insertion, deletion)

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *next;
};
struct node *head = NULL;
/* Insert at end */
void insert(int x)
{
    struct node *newnode, *temp;
    newnode = (struct node*)malloc(sizeof(struct node));
    newnode->data = x;
    newnode->next = NULL;

    if (head == NULL)
```

```
        head = newnode;
    else
    {
        temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newnode;
    }
}
void delete()
{
    struct node *temp;

    if (head == NULL)
    {
        printf("List is Empty\n");
        return;
    }
    temp = head;
    head = head->next;
    printf("Deleted element: %d\n", temp->data);
    free(temp);
}

void traverse()
{
    struct node *temp;

    if (head == NULL)
    {
        printf("List is Empty\n");
        return;
    }
    printf("Linked List: ");
    temp = head;
    while (temp != NULL)
    {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
int main()
{
    int n, value;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter elements:\n");
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &value);
        insert(value);
    }
```

```
    traverse();   // traversal
    delete();     // deletion
    traverse();   // traversal after deletion
    fflush(stdout);         // flush output buffer
    system("getmac");       // works only on Windows

    return 0;
}
```

Output :

```
Enter number of elements: 5
Enter elements:
3
1
2
4
6
Linked List: 3 -> 1 -> 2 -> 4 -> 6 -> NULL
Deleted element: 3
Linked List: 1 -> 2 -> 4 -> 6 -> NULL

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 13.257 s
Press any key to continue.
```

PROGRAM 12:

Linked list using Stacks

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

struct node
{
    int data;
    struct node *next;
};

struct node *top = NULL;

/* Push operation */
void push(int x)
{
    struct node *newnode;
    newnode = (struct node*)malloc(sizeof(struct node));
    newnode->data = x;
    newnode->next = top;
    top = newnode;
```

```
}

/* Pop operation */
void pop()
{
   struct node *temp;

   if (top == NULL)
   {
      printf("Stack is Empty\n");
      return;
   }

   temp = top;
   top = top->next;
   printf("Popped element: %d\n", temp->data);
   free(temp);
}

/* Display stack */
void display()
{
   struct node *temp;

   if (top == NULL)
   {
      printf("Stack is Empty\n");
      return;
   }

   printf("Stack elements: ");
   temp = top;
   while (temp != NULL)
   {
      printf("%d -> ", temp->data);
      temp = temp->next;
   }
   printf("NULL\n");
}

int main()
{
   int n, value;

   printf("Enter number of elements: ");
   scanf("%d", &n);

   printf("Enter elements:\n");
   for (int i = 0; i < n; i++)
   {
      scanf("%d", &value);
      push(value);
   }

   display();
```

```
    pop();
    display();

    fflush(stdout);         // flush output buffer
    system("getmac");       // works only on Windows

    return 0;
}
```

Output:

```
Enter number of elements: 3
Enter elements:
5
6
4
Stack elements: 4 -> 6 -> 5 -> NULL
Popped element: 4
Stack elements: 6 -> 5 -> NULL

Physical Address    Transport Name
=================== ==========================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 31.410 s
Press any key to continue.
```

PROGRAM 13:

Linked list using Queues

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

struct node
{
    int data;
    struct node *next;
};

struct node *front = NULL;
struct node *rear = NULL;

/* Enqueue operation */
void enqueue(int x)
{
    struct node *newnode;
    newnode = (struct node*)malloc(sizeof(struct node));
    newnode->data = x;
    newnode->next = NULL;
```

```
    if (rear == NULL)
    {
        front = rear = newnode;
    }
    else
    {
        rear->next = newnode;
        rear = newnode;
    }
}

/* Dequeue operation */
void dequeue()
{
    struct node *temp;

    if (front == NULL)
    {
        printf("Queue is Empty\n");
        return;
    }

    temp = front;
    front = front->next;

    printf("Deleted element: %d\n", temp->data);
    free(temp);

    if (front == NULL)
        rear = NULL;
}

/* Display queue */
void display()
{
    struct node *temp;

    if (front == NULL)
    {
        printf("Queue is Empty\n");
        return;
    }

    printf("Queue elements: ");
    temp = front;
    while (temp != NULL)
    {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main()
{
```

```
    int n, value;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter elements:\n");
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &value);
        enqueue(value);
    }

    display();
    dequeue();
    display();

    fflush(stdout);         // flush output buffer
    system("getmac");       // works only on Windows

    return 0;
}
```

Output:

```
Enter number of elements: 4
Enter elements:
1 2 3 4
Queue elements: 1 -> 2 -> 3 -> 4 -> NULL
Deleted element: 1
Queue elements: 2 -> 3 -> 4 -> NULL

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 12.604 s
Press any key to continue.
```

PROGRAM 14:

Time taken for arrays and linked lists during insertion and deletion

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

void arrayOperations(int n)
{
   int arr[10000];
   clock_t start, end;

   /* Insertion */
   start = clock();
   for (int i = 0; i < n; i++)
      arr[i] = i;
   end = clock();

   printf("Array Insertion Time: %f seconds\n",
         (double)(end - start) / CLOCKS_PER_SEC);

   /* Deletion */
   start = clock();
   for (int i = 0; i < n - 1; i++)
      arr[i] = arr[i + 1];
   end = clock();

   printf("Array Deletion Time: %f seconds\n",
         (double)(end - start) / CLOCKS_PER_SEC);
}

/* ---------- LINKED LIST ---------- */
struct node
{
   int data;
   struct node *next;
};

void linkedListOperations(int n)
{
   struct node *head = NULL, *temp;
   clock_t start, end;

   /* Insertion */
   start = clock();
   for (int i = 0; i < n; i++)
   {
      struct node *newnode =
         (struct node*)malloc(sizeof(struct node));
      newnode->data = i;
      newnode->next = head;
      head = newnode;
   }
```

```
    end = clock();

    printf("Linked List Insertion Time: %f seconds\n",
           (double)(end - start) / CLOCKS_PER_SEC);

    /* Deletion */
    start = clock();
    temp = head;
    if (temp != NULL)
    {
        head = head->next;
        free(temp);
    }
    end = clock();

    printf("Linked List Deletion Time: %f seconds\n",
           (double)(end - start) / CLOCKS_PER_SEC);
}

/* ---------- MAIN ---------- */
int main()
{
    int n;
    printf("Enter number of elements: ");
    scanf("%d", &n);

    arrayOperations(n);
    linkedListOperations(n);

    fflush(stdout);         // flush output buffer
    system("getmac");       // works only on Windows

    return 0;
}
```

Output :

```
Enter number of elements: 6200
Array Insertion Time: 0.000000 seconds
Array Deletion Time: 0.000000 seconds
Linked List Insertion Time: 0.016000 seconds
Linked List Deletion Time: 0.000000 seconds

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 4.366 s
Press any key to continue.
```

PROGRAM 15:
Sparse Matrix

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

int main()
{
   int r, c, i, j;
   int a[10][10];
   int triplet[20][3];
   int count = 0, k = 1;

   printf("Enter number of rows and columns: ");
   scanf("%d %d", &r, &c);

   printf("Enter matrix elements:\n");
   for (i = 0; i < r; i++)
      for (j = 0; j < c; j++)
         scanf("%d", &a[i][j]);

   /* Count non-zero elements */
   for (i = 0; i < r; i++)
      for (j = 0; j < c; j++)
         if (a[i][j] != 0)
            count++;

   /* First row of triplet */
   triplet[0][0] = r;
   triplet[0][1] = c;
   triplet[0][2] = count;

   /* Store non-zero elements */
   for (i = 0; i < r; i++)
   {
      for (j = 0; j < c; j++)
      {
         if (a[i][j] != 0)
         {
            triplet[k][0] = i;
            triplet[k][1] = j;
            triplet[k][2] = a[i][j];
            k++;
         }
      }
   }

   printf("\nSparse Matrix in Triplet Form:\n");
   printf("Row  Col  Value\n");
   for (i = 0; i <= count; i++)
      printf("%d    %d    %d\n",
           triplet[i][0],
           triplet[i][1],
           triplet[i][2]);
```

```
    fflush(stdout);         // flush output buffer
    system("getmac");       // works only on Windows

    return 0;
}
```

Output :

```
Enter number of rows and columns: 3 3
Enter matrix elements:
0 0 3
0 4 0
5 0 0

Sparse Matrix in Triplet Form:
Row  Col  Value
3    3    3
0    2    3
1    1    4
2    0    5

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 21.100 s
Press any key to continue.
```

PROGRAM 16:

Polynomial Representation

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

struct node
{
    int coeff;
    int power;
    struct node *next;
};

struct node *head = NULL;

/* Insert term at end */
void insert(int c, int p)
{
    struct node *newnode, *temp;
    newnode = (struct node*)malloc(sizeof(struct node));
    newnode->coeff = c;
    newnode->power = p;
    newnode->next = NULL;
```

```
    if (head == NULL)
        head = newnode;
    else
    {
        temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newnode;
    }
}

/* Display polynomial */
void display()
{
    struct node *temp = head;

    printf("Polynomial: ");
    while (temp != NULL)
    {
        printf("%dx^%d", temp->coeff, temp->power);
        if (temp->next != NULL)
            printf(" + ");
        temp = temp->next;
    }
    printf("\n");
}

int main()
{
    int n, c, p;

    printf("Enter number of terms: ");
    scanf("%d", &n);

    printf("Enter coefficient and power:\n");
    for (int i = 0; i < n; i++)
    {
        scanf("%d %d", &c, &p);
        insert(c, p);
    }

    display();

    fflush(stdout);         // flush output buffer
    system("getmac");       // works only on Windows

    return 0;
}
```

Output :

```
Enter number of terms: 3
Enter coefficient and power:
5 2
3 1
7 0
Polynomial: 5x^2 + 3x^1 + 7x^0

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 28.804 s
Press any key to continue.
```

PROGRAM 17:

How to create a tree

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

struct node {
   int data;
   struct node *left, *right;
};

struct node* insert(struct node* root, int val) {
   if (!root) {
      struct node* n = malloc(sizeof(struct node));
      n->data = val; n->left = n->right = NULL;
      return n;
   }
   if (val < root->data) root->left = insert(root->left, val);
   else root->right = insert(root->right, val);
   return root;
}

void inorder(struct node* root) {
   if (root) {
      inorder(root->left);
      printf("%d ", root->data);
      inorder(root->right);
   }
}

int main() {
```

```
    struct node* root = NULL;
    int n, v;

    printf("Enter the no. of ele: ");
    scanf("%d", &n);

    printf("Enter the elements: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &v);
        root = insert(root, v);
    }

    printf("Inorder Traversal: ");
    inorder(root);

    printf("\n");
    system("getmac");
    return 0;
}
```

Output:

```
Enter the no. of ele: 5
Enter the elements: 2 3 4 1 0
Inorder Traversal: 0 1 2 3 4

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 7.731 s
Press any key to continue.
```

PROGRAM 18

Using array we can construct a tree

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int tree[100];
    int n, i;
    printf("Enter number of nodes: ");
    scanf("%d", &n);
    printf("Enter tree elements:\n");
    for (i = 0; i < n; i++)
        scanf("%d", &tree[i]);
    printf("\nBinary Tree (Array Representation):\n");
    for (i = 0; i < n; i++)
```

```
    {
        printf("Node %d : %d", i, tree[i]);
        if (2*i + 1 < n)
            printf("  Left Child: %d", tree[2*i + 1]);
        if (2*i + 2 < n)
            printf("  Right Child: %d", tree[2*i + 2]);
        printf("\n");
    }
    fflush(stdout);         // flush output buffer
    system("getmac");
    return 0;
}
```

Output:

```
Enter number of nodes: 5
Enter tree elements:
1 2 3 4 5

Binary Tree (Array Representation):
Node 0 : 1  Left Child: 2  Right Child: 3
Node 1 : 2  Left Child: 4  Right Child: 5
Node 2 : 3
Node 3 : 4
Node 4 : 5

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 12.815 s
Press any key to continue.
```

PROGRAM 19:

Construct a binary tree using queues

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

/* -------- Tree Node -------- */
struct node
{
    int data;
    struct node *left;
    struct node *right;
};

/* -------- Queue -------- */
struct queue
{
    struct node *data;
    struct queue *next;
```

```
};

struct queue *front = NULL, *rear = NULL;

/* Enqueue */
void enqueue(struct node *x)
{
   struct queue newq = (struct queue)malloc(sizeof(struct queue));
   newq->data = x;
   newq->next = NULL;

   if (rear == NULL)
      front = rear = newq;
   else
   {
      rear->next = newq;
      rear = newq;
   }
}

/* Dequeue */
struct node* dequeue()
{
   struct queue *temp;
   struct node *x;

   if (front == NULL)
     return NULL;
   temp = front;
   x = temp->data;
   front = front->next;

   if (front == NULL)
      rear = NULL;
   free(temp);
   return x;
}
/* Create Tree Node */
struct node* createNode(int x)
{
   struct node newnode = (struct node)malloc(sizeof(struct node));
   newnode->data = x;
   newnode->left = newnode->right = NULL;
   return newnode;
}
/* Build Tree */
struct node* buildTree(int n)
{
   struct node *root, *temp;
   int value;
   printf("Enter root value: ");
   scanf("%d", &value);
   root = createNode(value);
   enqueue(root);
   for (int i = 1; i < n; i += 2)
```

```
    {
        temp = dequeue();
        printf("Enter left child of %d: ", temp->data);
        scanf("%d", &value);
        temp->left = createNode(value);
        enqueue(temp->left);

        if (i + 1 < n)
        {
            printf("Enter right child of %d: ", temp->data);
            scanf("%d", &value);
            temp->right = createNode(value);
            enqueue(temp->right);
        }
    }
    return root;
}
/* Inorder Traversal */
void inorder(struct node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}
/* -------- MAIN -------- */
int main()
{
    int n;
    struct node *root;
    printf("Enter total number of nodes: ");
    scanf("%d", &n);
    root = buildTree(n);
    printf("\nInorder Traversal:\n");
    inorder(root);
    printf("\n");
    fflush(stdout);         // flush output buffer
    system("getmac");

    return 0;
}
```

Output:

```
Enter total number of nodes: 5
Enter root value: 10
Enter left child of 10: 20
Enter right child of 10: 30
Enter left child of 20: 40
Enter right child of 20: 50

Inorder Traversal:
40 20 50 10 30

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 44.687 s
Press any key to continue.
```

PROGRAM 20:

Insertion of nodes

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

/* Tree Node */
struct node
{
   int data;
   struct node *left;
   struct node *right;
};

/* Queue for tree nodes */
struct queue
{
   struct node *data;
   struct queue *next;
};

struct queue *front = NULL, *rear = NULL;

/* Queue functions */
void enqueue(struct node *x)
{
   struct queue newq = (struct queue)malloc(sizeof(struct queue));
   newq->data = x;
   newq->next = NULL;

   if (rear == NULL)
      front = rear = newq;
```

```
    else
    {
        rear->next = newq;
        rear = newq;
    }
}

struct node* dequeue()
{
    struct queue *temp;
    struct node *x;

    if (front == NULL)
        return NULL;

    temp = front;
    x = temp->data;
    front = front->next;

    if (front == NULL)
        rear = NULL;

    free(temp);
    return x;
}

/* Create Node */
struct node* createNode(int value)
{
    struct node *newnode =
        (struct node*)malloc(sizeof(struct node));
    newnode->data = value;
    newnode->left = newnode->right = NULL;
    return newnode;
}

/* Insert Node */
void insertNode(struct node **root, int value)
{
    struct node *temp, *newnode;
    newnode = createNode(value);

    if (*root == NULL)
    {
        *root = newnode;
        return;
    }

    enqueue(*root);

    while (front != NULL)
    {
        temp = dequeue();

        if (temp->left == NULL)
```

```
      {
         temp->left = newnode;
         return;
      }
      else
         enqueue(temp->left);

      if (temp->right == NULL)
      {
         temp->right = newnode;
         return;
      }
      else
         enqueue(temp->right);
   }
}

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

/* MAIN */
int main()
{
   struct node *root = NULL;
   int n, value;

   printf("Enter number of nodes to insert: ");
   scanf("%d", &n);

   printf("Enter values:\n");
   for (int i = 0; i < n; i++)
   {
      scanf("%d", &value);
      insertNode(&root, value);
   }

   printf("Inorder Traversal after insertion:\n");
   inorder(root);
   printf("\n");

   fflush(stdout);         // flush output buffer
   system("getmac");

   return 0;
}
```

Output:

```
Enter number of nodes to insert:
5
Enter values:
1 2 3 4 5
Inorder Traversal after insertion:
4 2 5 1 3

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 15.270 s
Press any key to continue.
```

PROGRAM 21:
Deletion of nodes

```c
#include <stdio.h>
#include <stdlib.h>
/* ---------- Tree Node ---------- */
struct node
{
   int data;
   struct node *left;
   struct node *right;
};
/* ---------- Queue ---------- */
struct queue
{
   struct node *data;
   struct queue *next;
};
struct queue *front = NULL, *rear = NULL;
/* Queue operations */
void enqueue(struct node *x)
{
   struct queue *q = (struct queue*)malloc(sizeof(struct queue));
   q->data = x;
   q->next = NULL;

   if (rear == NULL)
      front = rear = q;
   else
   {
      rear->next = q;
      rear = q;
   }
```

```
}
struct node* dequeue()
{
   struct queue *temp;
   struct node *x;
   if (front == NULL)
      return NULL;
   temp = front;
   x = temp->data;
   front = front->next;
   if (front == NULL)
      rear = NULL;
   free(temp);
   return x;
}
/* Create Node */
struct node* createNode(int x)
{
   struct node *newnode =
      (struct node*)malloc(sizeof(struct node));
   newnode->data = x;
   newnode->left = newnode->right = NULL;
   return newnode;
}
/* Insert node (level order) */
void insert(struct node **root, int x)
{
   struct node *temp, *newnode = createNode(x);
   if (*root == NULL)
   {
      *root = newnode;
      return;
   }
   enqueue(*root);
   while (front != NULL)
   {
      temp = dequeue();
      if (temp->left == NULL)
      {
         temp->left = newnode;
         return;
      }
      else
         enqueue(temp->left);

      if (temp->right == NULL)
      {
         temp->right = newnode;
         return;
      }
      else
         enqueue(temp->right);
```

```
    }
  }
  /* Delete deepest node */
  void deleteDeepest(struct node *root, struct node *dnode)
  {
    struct node *temp;
    enqueue(root);
    while (front != NULL)
    {
      temp = dequeue();
      if (temp->left)
      {
        if (temp->left == dnode)
        {
          free(temp->left);
          temp->left = NULL;
          return;
        }
        else
          enqueue(temp->left);
      }
      if (temp->right)
      {
        if (temp->right == dnode)
        {
          free(temp->right);
          temp->right = NULL;
          return;
        }
        else
          enqueue(temp->right);
      }
    }
  }
  /* Delete given key */
  void deleteNode(struct node *root, int key)
  {
    struct node *temp, *keyNode = NULL;
    enqueue(root);
    while (front != NULL)
    {
      temp = dequeue();
      if (temp->data == key)
        keyNode = temp;
      if (temp->left) enqueue(temp->left);
      if (temp->right) enqueue(temp->right);
    }

    if (keyNode != NULL)
    {
      int x = temp->data;
      deleteDeepest(root, temp);
```

```
        keyNode->data = x;
        printf("Node deleted successfully\n");
    }
    else
        printf("Node not found\n");
}

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

int main()
{
    struct node *root = NULL;
    int n, value, key;
    printf("Enter number of nodes: ");
    scanf("%d", &n);
    printf("Enter elements:\n");
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &value);
        insert(&root, value);
    }
    printf("Enter element to delete: ");
    scanf("%d", &key);
    deleteNode(root, key);
    printf("Inorder traversal after deletion:\n");
    inorder(root);
    printf("\n");
    fflush(stdout);         // flush output buffer
    system("getmac");
    return 0;
}
```

Output:

```
Enter number of nodes: 5
Enter elements:
1 2 3 4 5
Enter element to delete: 2
Node deleted successfully
Inorder traversal after deletion:
4 5 1 3

Physical Address    Transport Name
=================== ==========================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 21.612 s
Press any key to continue.
```

PROGRAM 22:
Depth search tree

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 20
int stack[MAX], top = -1;
void push(int x) {
   if(top < MAX-1)
      stack[++top] = x;
}
int pop() {
   if(top >= 0)
      return stack[top--];
   return -1;
}
int isEmpty() {
   return top == -1;
}

void DFS(int n, int adj[n][n], int start) {
   int visited[n];
   for(int i=0; i<n; i++) visited[i] = 0;
   push(start);
   printf("DFS Traversal: ");
   while(!isEmpty()) {
      int v = pop();
      if(!visited[v]) {
         printf("%d ", v);
         visited[v] = 1;
      }
      // Push adjacent nodes
      for(int i = n-1; i >=0; i--) {
         if(adj[v][i] && !visited[i])
            push(i);
      }
   }
   printf("\n");
```

```
}
int main() {
   int n, start;

   printf("Enter number of vertices: ");
   scanf("%d", &n);

   int adj[n][n];
   printf("Enter adjacency matrix:\n");
   for(int i=0;i<n;i++)
      for(int j=0;j<n;j++)
         scanf("%d",&adj[i][j]);

   printf("Enter starting vertex (0 to %d): ", n-1);
   scanf("%d",&start);

   DFS(n, adj, start);

   fflush(stdout);        // flush output buffer
   system("getmac");      // works only on Windows

   return 0;
}
```

Output:

```
Enter number of vertices: 4
Enter adjacency matrix:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
Enter starting vertex (0 to 3): 0
DFS Traversal: 0 1 3 2

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 31.817 s
Press any key to continue.
```

Program 23:
Breath search tree

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 20
int queue[MAX], front=-1, rear=-1;
void enqueue(int x) {
   if(rear < MAX-1) {
      if(front==-1) front=0;
      queue[++rear] = x;
```

```
    }
}
int dequeue() {
    if(front != -1) {
        int x = queue[front];
        if(front==rear) front=rear=-1;
        else front++;
        return x;
    }
    return -1;
}
int isEmpty() {
    return front==-1;
}
void BFS(int n, int adj[n][n], int start) {
    int visited[n];
    for(int i=0;i<n;i++) visited[i]=0;
    enqueue(start);
    visited[start]=1;
    printf("BFS Traversal: ");
    while(!isEmpty()) {
        int v = dequeue();
        printf("%d ", v);
        for(int i=0;i<n;i++) {
            if(adj[v][i] && !visited[i]) {
                enqueue(i);
                visited[i]=1;
            }
        }
    }
    printf("\n");
}
int main() {
    int n, start;
    printf("Enter number of vertices: ");
    scanf("%d",&n);
    int adj[n][n];
    printf("Enter adjacency matrix:\n");
    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
            scanf("%d",&adj[i][j]);

    printf("Enter starting vertex (0 to %d): ", n-1);
    scanf("%d",&start);
    BFS(n, adj, start);
    fflush(stdout);         // flush output buffer
    system("getmac");       // works only on Windows
    return 0;
}
```

Output:

```
Enter number of vertices: 4
Enter adjacency matrix:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
Enter starting vertex (0 to 3): 0
BFS Traversal: 0 1 2 3

Physical Address    Transport Name
=================== ==========================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 47.032 s
Press any key to continue.
```

PROGRAM 24:
Level order

```c
#include <stdio.h>
#include <stdlib.h>
/* ---------- Tree Node ---------- */
struct node {
   int data;
   struct node *left;
   struct node *right;
};
/* ---------- Queue for Tree Nodes ---------- */
struct queue {
   struct node *data;
   struct queue *next;
};
struct queue *front = NULL, *rear = NULL;
/* Queue Operations */
void enqueue(struct node *x) {
   struct queue *q = (struct queue*)malloc(sizeof(struct queue));
   q->data = x;
   q->next = NULL;
   if(rear == NULL)
      front = rear = q;
   else {
      rear->next = q;
      rear = q;
   }
}
struct node* dequeue() {
   if(front == NULL) return NULL;
   struct queue *temp = front;
   struct node *x = temp->data;
   front = front->next;
   if(front == NULL) rear = NULL;
```

```
    free(temp);
    return x;
}
int isEmpty() {
    return front == NULL;
}
/* Create Tree Node */
struct node* createNode(int val) {
    struct node *newnode = (struct node*)malloc(sizeof(struct node));
    newnode->data = val;
    newnode->left = newnode->right = NULL;
    return newnode;
}
/* Level Order Traversal */
void levelOrder(struct node *root) {
    if(root == NULL) return;
    enqueue(root);

    printf("Level Order Traversal: ");
    while(!isEmpty()) {
        struct node *temp = dequeue();
        printf("%d ", temp->data);
        if(temp->left) enqueue(temp->left);
        if(temp->right) enqueue(temp->right);
    }
    printf("\n");
}
/* Build Tree from User Input (Level Order) */
struct node* buildTree(int n) {
    if(n <= 0) return NULL;

    int value;
    printf("Enter value of root: ");
    scanf("%d", &value);
    struct node *root = createNode(value);
    enqueue(root);
    int count = 1;
    while(count < n) {
        struct node *temp = dequeue();

        // Left child
        if(count < n) {
            printf("Enter left child of %d: ", temp->data);
            scanf("%d", &value);
            temp->left = createNode(value);
            enqueue(temp->left);
            count++;
        }

        // Right child
        if(count < n) {
            printf("Enter right child of %d: ", temp->data);
```

```
            scanf("%d", &value);
            temp->right = createNode(value);
            enqueue(temp->right);
            count++;
        }
    }
    return root;
}
/* MAIN */
int main() {
    int n;
    printf("Enter number of nodes: ");
    scanf("%d", &n);

    struct node *root = buildTree(n);

    levelOrder(root);

    fflush(stdout);         // flush output buffer
    system("getmac");       // works only on Windows
    return 0;
}
```

Output:

```
Enter number of nodes: 5
Enter value of root: 10
Enter left child of 10: 20
Enter right child of 10: 30
Enter left child of 20: 40
Enter right child of 20: 50
Level Order Traversal: 30 40 50 10 20 30 40 50

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 14.619 s
Press any key to continue.
```

PROGRAM 25:
DFS and BFS using adjacency list(use stacks and queues in program)

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 20
/* Stack for DFS */
int stack[MAX], top = -1;
Void push(int x) {
   if(top < MAX-1) stack[++top] = x;
}
int pop() {
   if(top >= 0) return stack[top--];
   return -1;
}
int isStackEmpty() { return top == -1; }
/* Queue for BFS */
int queue[MAX], front=-1, rear=-1;
void enqueue(int x) {
   if(rear < MAX-1) {
      if(front == -1) front=0;
      queue[++rear] = x;
   }
}
int dequeue() {
   if(front != -1) {
      int x = queue[front];
      if(front==rear) front=rear=-1;
      else front++;
      return x;
   }
   return -1;
}
int isQueueEmpty() { return front==-1; }

/* Node for adjacency list */
struct adjNode {
   int vertex;
   struct adjNode *next;
};
/* Adjacency list array */
struct adjNode* adjList[MAX];
/* Add edge to adjacency list (undirected) */
void addEdge(int u, int v) {
   struct adjNode *newNode = (struct adjNode*)malloc(sizeof(struct adjNode));
   newNode->vertex = v;
   newNode->next = adjList[u];
   adjList[u] = newNode;
   // For undirected graph
   newNode = (struct adjNode*)malloc(sizeof(struct adjNode));
   newNode->vertex = u;
   newNode->next = adjList[v];
```

```
    adjList[v] = newNode;
}

/* DFS using stack */
void DFS(int n, int start) {
    int visited[n];
    for(int i=0;i<n;i++) visited[i]=0;
    push(start);
    printf("DFS Traversal: ");
    while(!isStackEmpty()) {
        int v = pop();
        if(!visited[v]) {
            printf("%d ", v);
            visited[v] = 1;
            struct adjNode *temp = adjList[v];
            // push adjacent vertices in reverse order for correct sequence
            int tempStack[MAX], t=0;
            while(temp) {
                if(!visited[temp->vertex])
                    tempStack[t++] = temp->vertex;
                temp = temp->next;
            }
            for(int i=t-1;i>=0;i--) push(tempStack[i]);
        }
    }
    printf("\n");
}

/* BFS using queue */
void BFS(int n, int start) {
    int visited[n];
    for(int i=0;i<n;i++) visited[i]=0;
    enqueue(start);
    visited[start]=1;
    printf("BFS Traversal: ");
    while(!isQueueEmpty()) {
        int v = dequeue();
        printf("%d ", v);
        struct adjNode *temp = adjList[v];
        while(temp) {
            if(!visited[temp->vertex]) {
                enqueue(temp->vertex);
                visited[temp->vertex]=1;
            }
            temp = temp->next;
        }
    }
    printf("\n");
}

int main() {
    int n, e, u, v, start;
```

```
    printf("Enter number of vertices: ");
    scanf("%d",&n);
    printf("Enter number of edges: ");
    scanf("%d",&e);
    for(int i=0;i<n;i++) adjList[i]=NULL;
    printf("Enter edges (u v):\n");
    for(int i=0;i<e;i++) {
        scanf("%d %d",&u,&v);
        addEdge(u,v);
    }
    printf("Enter starting vertex: ");
    scanf("%d",&start);
    DFS(n,start);
    BFS(n,start);
    fflush(stdout);         // flush output buffer
    system("getmac");       // works only on Windows

    return 0;
}
```

```
Enter number of vertices: 4
Enter number of edges: 4
Enter edges (u v):
0 1
0 2
1 3
2 3
Enter starting vertex: 0
DFS Traversal: 0 2 3 1
BFS Traversal: 0 2 1 3

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 24.563 s
Press any key to continue.
```

Program 26:
Calloc and malloc functions checking if junk/zero is initialized to them

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

int main() {
   int n;
   printf("Enter number of elements: ");
   scanf("%d", &n);

   int arr_malloc = (int)malloc(n * sizeof(int));
   int arr_calloc = (int)calloc(n, sizeof(int));

   printf("\nValues using malloc (usually junk):\n");
   for(int i=0;i<n;i++)
      printf("%d ", arr_malloc[i]);

   printf("\nValues using calloc (always zero):\n");
   for(int i=0;i<n;i++)
      printf("%d ", arr_calloc[i]);

   free(arr_malloc);
   free(arr_calloc);

   printf("\n");

   fflush(stdout);         // flush output buffer
   system("getmac");       // works only on Windows

   return 0;
}
```

Output:

```
Enter number of elements: 5

Values using malloc (usually junk):
8261096 8262672 83886085 36971 8261096
Values using calloc (always zero):
0 0 0 0 0

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 9.898 s
Press any key to continue.
```

PROGRAM 27:
Linked list circular basic operations

```c
#include <stdio.h>
#include <stdlib.h>
/* Node Structure */
struct Node {
   int data;
   struct Node *next;
};
/* Create Node */
struct Node* createNode(int val) {
   struct Node *newnode = (struct Node*)malloc(sizeof(struct Node));
   newnode->data = val;
   newnode->next = NULL;
   return newnode;
}
/* Insert at end */
struct Node* insertEnd(struct Node *head, int val) {
   struct Node *newnode = createNode(val);
   if(head == NULL) {
      newnode->next = newnode;
      return newnode;
   }
   struct Node *temp = head;
   while(temp->next != head) temp = temp->next;
   temp->next = newnode;
   newnode->next = head;
   return head;
}
/* Delete a node with value key */
struct Node* deleteNode(struct Node *head, int key) {
   if(head == NULL) return NULL;
   struct Node *curr = head, *prev = NULL;
   // Single node case
   if(head->data == key && head->next == head) {
      free(head);
      return NULL;
   }
   // Find the node to delete
   do {
      if(curr->data == key) break;
      prev = curr;
      curr = curr->next;
   } while(curr != head);
   if(curr->data != key) {
      printf("Node not found\n");
      return head;
   }
   // Deleting head node
   if(curr == head) {
      struct Node *last = head;
```

```
        while(last->next != head) last = last->next;
        last->next = head->next;
        head = head->next;
        free(curr);
    } else {
        prev->next = curr->next;
        free(curr);
    }
    return head;
}
/* Traverse */
void traverse(struct Node *head) {
    if(head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node *temp = head;
    printf("Circular Linked List: ");
    do {
        printf("%d ", temp->data);
        temp = temp->next;
    } while(temp != head);
    printf("\n");
}
/* MAIN */
int main() {
    struct Node *head = NULL;
    int n, val, key;
    printf("Enter number of elements: ");
    scanf("%d",&n);
    printf("Enter elements:\n");
    for(int i=0;i<n;i++) {
        scanf("%d",&val);
        head = insertEnd(head,val);
    }
    traverse(head);
    printf("Enter value to delete: ");
    scanf("%d",&key);
    head = deleteNode(head,key);
    traverse(head);
    fflush(stdout);         // flush output buffer
    system("getmac");       // works only on Windows
    return 0;
}
```

Output:

```
Enter number of elements: 5
Enter elements:
1 2 3 4 5
Circular Linked List: 1 2 3 4 5
Enter value to delete: 3
Circular Linked List: 1 2 4 5

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 18.645 s
Press any key to continue.
```

PROGRAM 28:
How to implement comparision of 2 strings using built in function?

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main() {
   char str1[100], str2[100];
   printf("Enter first string: ");
   scanf("%s", str1);
   printf("Enter second string: ");
   scanf("%s", str2);
   int result = strcmp(str1, str2);  // built-in function
   if(result == 0)
      printf("Strings are equal\n");
   else if(result < 0)
      printf("First string is smaller than second string\n");
   else
      printf("First string is greater than second string\n");
   fflush(stdout);         // flush output buffer
       system("getmac");
       return 0;
}
```

Output:

```
Enter first string: hello
Enter second string: world
First string is smaller than second string

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 12.011 s
Press any key to continue.
```

```
Enter first string: hello
Enter second string: world
First string is smaller than second string

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 12.011 s
Press any key to continue.
```

PROGRAM 29:

In linked list insertion in the middle and deletion in the middle

```c
#include <stdio.h>
#include <stdlib.h>
/* Node structure */
struct Node {
   int data;
   struct Node *next;
};
/* Insert at position (1-based) */
struct Node* insertMiddle(struct Node *head, int pos, int val) {
   struct Node *newnode = (struct Node*)malloc(sizeof(struct Node));
   newnode->data = val;
   newnode->next = NULL;
   if(pos == 1) {
      newnode->next = head;
      return newnode;
   }
   struct Node *temp = head;
   for(int i=1;i<pos-1 && temp != NULL;i++)
      temp = temp->next;
   if(temp == NULL) {
      printf("Position out of range\n");
      free(newnode);
      return head;
```

```
    }
    newnode->next = temp->next;
    temp->next = newnode;
    return head;
}
/* Delete at position (1-based) */
struct Node* deleteMiddle(struct Node *head, int pos) {
    if(head == NULL) return NULL;
    struct Node *temp = head, *prev = NULL;
    if(pos == 1) {
        head = head->next;
        free(temp);
        return head;
    }
    for(int i=1;i<pos && temp!=NULL;i++) {
        prev = temp;
        temp = temp->next;
    }
    if(temp == NULL) {
        printf("Position out of range\n");
        return head;
    }
    prev->next = temp->next;
    free(temp);
    return head;
}
/* Traverse */
void traverse(struct Node *head) {
    struct Node *temp = head;
    printf("Linked List: ");
    while(temp) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
/* MAIN */
int main() {
    struct Node *head = NULL;
    int n, val, pos;
    printf("Enter number of initial elements: ");
    scanf("%d",&n);
    printf("Enter elements:\n");
    for(int i=0;i<n;i++) {
        scanf("%d",&val);
        head = insertMiddle(head, i+1, val); // insert at end initially
    }
    traverse(head);
    // Insert in middle
    printf("Enter position to insert: ");
    scanf("%d",&pos);
```

```
    printf("Enter value to insert: ");
    scanf("%d",&val);
    head = insertMiddle(head,pos,val);
    traverse(head);
    // Delete in middle
    printf("Enter position to delete: ");
    scanf("%d",&pos);
    head = deleteMiddle(head,pos);
    traverse(head);
    fflush(stdout);        // flush output buffer
    system("getmac");      // works only on Windows
    return 0;
}
```

Output:

```
Enter number of initial elements: 4
Enter elements:
1 2 3 4
Linked List: 1 2 3 4
Enter position to insert: 3
Enter value to insert: 25
Linked List: 1 2 25 3 4
Enter position to delete: 2
Linked List: 1 25 3 4

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 32.957 s
Press any key to continue.
```

PROGRAM 30:

Binary tree traversal

```
#include <stdio.h>
#include <stdlib.h>
/* Node structure */
struct Node {
    int data;
    struct Node *left;
    struct Node *right;
};
/* Create new node */
struct Node* createNode(int val) {
    struct Node *newnode = (struct Node*)malloc(sizeof(struct Node));
    newnode->data = val;
```

```
    newnode->left = newnode->right = NULL;
    return newnode;
}
/* Build tree recursively (user input) */
struct Node* buildTree() {
    int val;
    printf("Enter value (-1 for NULL): ");
    scanf("%d", &val);
    if(val == -1) return NULL;
    struct Node *root = createNode(val);
    printf("Enter left child of %d:\n", val);
    root->left = buildTree();
    printf("Enter right child of %d:\n", val);
    root->right = buildTree();
    return root;
}

/* Preorder traversal */
void preorder(struct Node *root) {
    if(root == NULL) return;
    printf("%d ", root->data);
    preorder(root->left);
    preorder(root->right);
}
/* Inorder traversal */
void inorder(struct Node *root) {
    if(root == NULL) return;
    inorder(root->left);
    printf("%d ", root->data);
    inorder(root->right);
}
/* Postorder traversal */
void postorder(struct Node *root) {
    if(root == NULL) return;
    postorder(root->left);
    postorder(root->right);
    printf("%d ", root->data);
}
/* MAIN */
int main() {
    struct Node *root = buildTree();
    printf("Preorder Traversal: ");
    preorder(root);
    printf("\n");
    printf("Inorder Traversal: ");
    inorder(root);
    printf("\n");
    printf("Postorder Traversal: ");
    postorder(root);
    printf("\n");
    fflush(stdout);         // flush output buffer
    system("getmac");       // works only on Windows
```

```
    return 0;
}
```

```
Enter value (-1 for NULL): 10
Enter left child of 10:
Enter value (-1 for NULL): 20
Enter left child of 20:
Enter value (-1 for NULL): -1
Enter right child of 20:
Enter value (-1 for NULL): 30
Enter left child of 30:
Enter value (-1 for NULL): -1
Enter right child of 30:
Enter value (-1 for NULL): -1
Enter right child of 10:
Enter value (-1 for NULL): 40
Enter left child of 40:
Enter value (-1 for NULL): -1
Enter right child of 40:
Enter value (-1 for NULL): -1
Preorder Traversal: 10 20 30 40
Inorder Traversal: 20 30 10 40
Postorder Traversal: 30 20 40 10

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 36.400 s
Press any key to continue.
```

PROGRAM 31:

Check for modulo division , queue full and queue empty for circular queues.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
int cq[MAX];
int front = -1, rear = -1;
/* Check if queue is full */
int isFull() {
    return (rear + 1) % MAX == front;
}
/* Check if queue is empty */
int isEmpty() {
    return front == -1;
}
/* Enqueue */
```

```
void enqueue(int val) {
   if(isFull()) {
      printf("Queue is full! Cannot insert %d\n", val);
      return;
   }
   if(front == -1) front = 0;
   rear = (rear + 1) % MAX;
   cq[rear] = val;
   printf("%d inserted\n", val);
}
/* Dequeue */
int dequeue() {
   if(isEmpty()) {
      printf("Queue is empty! Cannot delete\n");
      return -1;
   }
   int val = cq[front];
   if(front == rear) {
      front = rear = -1;  // only one element was present
   } else {
      front = (front + 1) % MAX;
   }
   return val;
}
/* Display queue */
void display() {
   if(isEmpty()) {
      printf("Queue is empty\n");
      return;
   }
   printf("Queue elements: ");
   int i = front;
   while(1) {
      printf("%d ", cq[i]);
      if(i == rear) break;
      i = (i + 1) % MAX;
   }
   printf("\n");
}
int main() {
   int choice, val;

   do {
      printf("\n1.Enqueue 2.Dequeue 3.Display 4.Exit\nEnter choice: ");
      scanf("%d",&choice);
      switch(choice) {
         case 1:
            printf("Enter value to insert: ");
            scanf("%d",&val);
            enqueue(val);
            break;
         case 2:
```

```
            val = dequeue();
            if(val != -1) printf("Deleted: %d\n", val);
            break;
        case 3:
            display();
            break;
        case 4:
            break;
        default:
            printf("Invalid choice\n");
    }
  } while(choice != 4);

  fflush(stdout);        // flush output buffer
  system("getmac");      // works only on Windows

  return 0;
}
```

Output:

```
1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice: 1
Enter value to insert: 10
10 inserted

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice: 1
Enter value to insert: 20
20 inserted

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice: 3
Queue elements: 10 20

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice: 2
Deleted: 10

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice: 3
Queue elements: 20

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice: 4

Physical Address    Transport Name
=================== ============================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 46.631 s
Press any key to continue.
```

PROGRAM 32:

Sparse matrix using linked lists.

```
#include <stdio.h>
#include <stdlib.h>
/* Node for non-zero element */
struct Node {
   int row;
   int col;
   int value;
   struct Node *next;
};
/* Create new node */
struct Node* createNode(int r, int c, int val) {
   struct Node *newnode = (struct Node*)malloc(sizeof(struct Node));
   newnode->row = r;
   newnode->col = c;
   newnode->value = val;
   newnode->next = NULL;
   return newnode;
}
/* Insert node at end */
struct Node* insertNode(struct Node *head, int r, int c, int val) {
   struct Node *newnode = createNode(r,c,val);
   if(head == NULL) return newnode;
   struct Node *temp = head;
   while(temp->next != NULL) temp = temp->next;
   temp->next = newnode;
   return head;
}
/* Display sparse matrix */
void displaySparse(struct Node *head, int rows, int cols) {
   struct Node *temp = head;
   printf("\nSparse Matrix:\n");
   for(int i=0;i<rows;i++){
      for(int j=0;j<cols;j++){
         if(temp != NULL && temp->row==i && temp->col==j){
            printf("%d ", temp->value);
            temp = temp->next;
         } else {
            printf("0 ");
         }
      }
      printf("\n");
   }
}
int main() {
   int rows, cols, n, r, c, val;
   struct Node *head = NULL;
   printf("Enter number of rows: ");
   scanf("%d",&rows);
```

```
    printf("Enter number of columns: ");
    scanf("%d",&cols);

    printf("Enter number of non-zero elements: ");
    scanf("%d",&n);
    printf("Enter row, column and value of each non-zero element:\n");
    for(int i=0;i<n;i++){
        scanf("%d %d %d",&r,&c,&val);
        head = insertNode(head,r,c,val);
    }
    displaySparse(head, rows, cols);
    fflush(stdout);         // flush output buffer
    system("getmac");       // works only on Windows
    return 0;
}
```

Output:

```
Enter number of rows: 3
Enter number of columns: 3
Enter number of non-zero elements: 4
Enter row, column and value of each non-zero element:
0 0 5
0 2 8
1 1 3
2 0 6

Sparse Matrix:
5 0 8
0 3 0
6 0 0

Physical Address    Transport Name
=================== ==========================================================
0A-00-27-00-00-16   \Device\Tcpip_{EBEDA09F-6BE9-4192-AB87-F35828274952}
1C-BF-C0-9F-F8-41   \Device\Tcpip_{4993E623-2001-4053-ACC6-CC8A9BF04E62}
84-2A-FD-4A-CD-12   Media disconnected

Process returned 0 (0x0)   execution time : 31.649 s
Press any key to continue.
```