Experiment 01(a)

Aim: Write a program to implement linear search in an array.

Software Used: Turbo C.

Source Code

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int a[10], i, c, item, Flag=0;
   clrscr();
   printf("Enter how many elements you want in an array: ");
   scanf("%d", &c);
   for(i=0; i<=0; i++)
   {
      scanf("%d", &a[i]);
   }
   printf("Enter the element to be Searched: ");
   scanf("%d", &item);
   for(i=0; i<=c; i++)
   {
     if(a[i] == item)
```

```
        {
            printf("Element found at position %d", i);
            Flag = 1;
            break;
        }
    }
    if (Flag == 0)
        {
            printf("Element not found");
        }
    getch();
}
```

Experiment 01(b)

Aim: To implement binary search in an array using function.

Source Code

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a[10], i, c, item, flag=0, beg, mid, last;
    printf("Enter how many elements you want to
            enter in an array: ");
    scanf("%d", &c);
    printf("Enter elements of array:");
    scanf("%d",
    for(i=0; i<=c; i++)
    {
        scanf("%d", &a[i]);
    }

    beg=0;
    last = c-1;
    printf("Enter the elements to be searched:");
    scanf("%d", &item);
    while(beg<=last)
    {
```

```
mid = (last + beg) / 2;
if (item = a[mid])
{
    printf("Element found at pos : %d", mid+1);
    flag = 1;
    break;
    }
else if(item < a[mid])
    {
    last = mid - 1;
    }
else
    {
    beg = mid + 1;
    }
}
if(flag == 0)
{
    printf("Element not found!");
    }
getch();
}
```

14/7

21/8/17

CLASSTIME | Page No.
Date / /

Experiment 2

Aim: To implement sparse matrix using Array & Linked List.

Software Used: Turbo C

Using Array – Source code:

```
#include<stdio.h>
#include<conio.h>
  void main()
  {
    int arr1[15][15], arr2[15][15], n, i, j, r, c=0,
    c1=0, c2=0;
    clrscr();
    printf(" Enter no. of Rows: ");
    scanf("%d", &r);
    printf(" Enter no. of Columns: ");
    scanf("%d", &c);
    printf(" Enter the elements :");
    for(i=0; i<r; i++)
    {
      {
        scanf("%d", &arr1[i][j]);
```

```
            if (arr1[i][j] == 0)
                c1++
            else
                c2++;
        }
    }
    if (c1 > c2)
    {
        printf("The matrix is a sparse matrix.");
        for(i=0; i<r; i++)
        {
            for(j=0; j<c; j++)
                if (arr1[i][j] != 0)
                {
                    arr2[0][k] = i;
                    arr2[1][k] = j;
                    arr2[2][k] = arr1[i][j];
                    k++;
                }
        }
    }
    for(i=0; i<r; i++)
    {
        for(j=0; j<k; j++)
        {
            printf("%d", arr2[i][j]);
        }
    }
```

```
        printf("\n");
    }
}
else
    printf("The matrix is not a Sparse matrix!");
    getch();
}
```

- **Output**

```
Enter no. of Rows : 3
Enter no. of Columns : 3
Enter elements : 1 0 0 3 0 0 6 7 0
The matrix is a Sparse matrix.
0   1   2   2
0   0   1   1
1   3   6   7
```

(b) **Using Linked List**

```
#include <stdio.h>
#include <conio.h>
  struct node
  {
    int row; int column; int value;
    struct node *next;
  };
```

CLASSTIME | Page No.
Date / /

```
void Umaker(struct node **start, int row, int col,
                int value)
{
   struct node *temp, *r;
   temp =*start;
   if (temp == NULL)
   {
      temp = (struct node *)malloc (sizeof(struct node));
      temp -> row = row;
      temp -> col = col;
      temp -> value = value;
      *start = temp;
      temp -> next = NULL;
   }
else
   {
     while (temp -> next != NULL)
     {
       temp = temp -> next;
     }
r = (struct node *) malloc (sizeof (struct node));
r -> row = row;
r -> col = col;
r -> value = value;
r -> next = NULL;
```

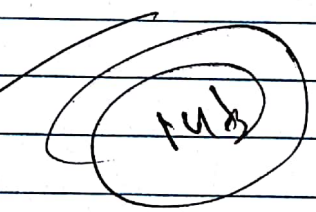
```
        temp -> next = r;
        }
    }

void printlist (struct node *start)
{
    struct node * temp;
    temp = start;
    printf ("ROW \t COL \t VALUE \n");
    while (temp != NULL)
    {
        printf ("%d" \t %d \t %d \n", temp -> row,
                temp -> col; temp -> value);
        temp = temp -> next;
    }
}

void main()
{
    int i=0, j=0, row, col, count;
    int arr[30] [30];
    struct node *start;
    start = NULL;
    clrscr();
    printf ("Enter no. of rows: ");
    scanf ("%d", &row);
    printf ("Enter no. of columns: ");
    scanf ("%d", &col);
```

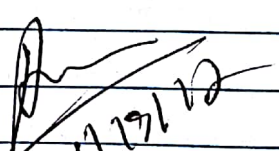
```
printf ("Enter the elements : ");
for(i=0; i<row; i++)
{
  for(j=0; j<col; j++)
  {
    scanf ("%d", &arr[i][j]);
  }
}
for(i=0; j<row; i++)
  {
    for(j=0; j<col; j++)
      {
        if(arr[i][j]==0)
        count++;
      }
}
if (count >= (row * col)/2)
  {
  printf ("The matrix is a sparse matrix!");
  printf ("The efficient linked list is : \n");
  for (i=0; i<row; i++)
  {
    for (j=0; j<col; j++)
      {
        if (arr[i][j]=0)
        llmaker (&start, i, j, arr[i][j]);
      }
```

```
    printlist(start);
  }
else
    printf("The matrix is not a sparse matrix !! ");
    getch();
}
```

Output

```
Enter no. of rows : 3
Enter no. of columns : 3
Enter the elements : 1 0 0 3 0 0 6 5 0
The matrix is a sparse matrix.
The Efficient linked list is
Row   COL   VALUE
 0     0     1
 1     0     3
 2     0     6
 2     1     5
```





Telco Dt.: Pg.:

Experiment 3

- **Aim**: Linked List to store student's data & perform operations.
- **Software Used**: TURBO C.
- **Source Code**:

```
#include <stdio.h>
#include <stdlib.h>
struct student{
        char name[20];
        int roll[10];
        int clas;
        struct node *next;
};
struct student *first=NULL, *last=NULL, *k;
void create(int n)
{
  int i;
  first = (struct student*) malloc(sizeof(struct student));
  printf("Enter the first name of student: ");
  scanf("%s", first->name);
  printf("Enter the roll number: ");
  scanf("%d", first->roll);
  first -> next = NULL;
  last = first;
```

Telco Dt.: Pg.:

```
for (i=1; i<n; i++)
{
  k = (struct student *) malloc (sizeof (struct student));
  printf("\n Enter first name of the student : ");
  scanf ("%.s", k -> name);
  printf ("\n Enter the roll no. of the student : ");
  scanf ("%.s" k -> roll);
  k -> next = NULL;
  last -> next = k;
  last = k;
  }
}
void display()
{
  struct student * t;
  t = first;
  while (t != NULL){
    printf ("\n Roll no. of the student : %.s", t -> roll);
    printf ("\n First name of the student : %.s", t -> name);
    t = t -> next;
    }
}
void insertafter ()
{
  int r[10];
  int flag = 0;
  printf ("\n Enter next roll number : ");
  scanf ("%d", r);
```

```
struct student *t;
t = first;
while (t != NULL)
{
  if (strcmpi(r, t->roll) == 0)
  {
    k = (struct student *) malloc (sizeof(struct student));
    printf ("\n Enter the first name of student : ");
    scanf ("%s", k->name);
    printf ("\n Enter the roll no. of the student : ");
    scanf ("%s", k->roll);
    k->next = t->next;
    t->next = k;
    flag = 1;
    break;
  }
  t = t->next;
}
if (flag == 0)
  printf ("\n Element not found !!! ");
}
void del()
{
  struct student *back, *t, *k;
  char r[10];
  int flag = 0;
  printf ("\n Enter roll no. to delete : ");
  scanf ("%s", r);
```

```
if (strcmpi(r, first -> roll) == 0)
{
  first = first -> next;
  flag = 1;
}
else
  {
    back = first;
    k = first -> next;
    while (k != NULL)
    {
      if (strcmpi(r, k -> roll) == 0)
      {
        back -> next = k -> next;
        flag = 1;
        break;
      }
    }
  }
if (flag == 0)
printf("\n The element not found!!!");
}
void reverse(){
struct student *q, r, *s;
  q = first;
  r = NULL;
  while(q != NULL){
```

```
        s = r;
        r = q;
        q = q -> next;
        r => next = s;
    }
    first = r;
}
int main()
{
    int n, o;
    while(o != 0)
    {
     printf("\n MENU\n");
     print("\n1. Create Database");
     printf("\n2. Display Database");
     printf("\n3. Insert a record");
     printf("\n4. Delete a record.");
     printf("\n5. Reverse the list");
     printf("\n0. Exit");
     scanf("%d", &o);
     switch(o){
     case 1: printf("\n Enter max. size of database: ");
     scanf("%d", &n);
     create(n); break; case 2: display(); break;
     case 3: insertafter(); break;
     case 4: del(); break;
     case 5: reverse(); break; case 0: exit(0); break;
     default: printf("\n Wrong choice !!"); }}
}
```

Telco Dt.: Pg.:

Experiment 4

- **Aim**: WAP to create a doubly linked list & perform operations like insertion, deletion and append.
- **Software Used**: Turbo C.
- **Source Code**

```
#include <stdio.h>
#include <stdlib.h>
struct dnode
{
   struct dnode *prev;
   int data;
   struct dnode *next;
};
struct dnode * start = NULL;
void append (int);
void insert (int);                    // at beginning
void remove(int);
void display();
int main ()
{
   int n, ch;
   do
   { printf ("\n Operations on doubly linked list ");
     printf("\n1. Append \n2. Insert \n3. Remove
        \n4. Display  \n0. Exit\n ");
```

```
printf("Enter your choice : ");
scanf("%d", &ch);
switch(ch)
  {
    case 1: printf("Enter number : ");
    scanf("%d", &n);
    append(n); break;
    case 2: printf("\n Enter number : ");
    scanf("%d", &n);
    insert(n);
    break;
    case 3: printf("\n Enter number to delete : ");
    scanf("%d", &n);
    remove(n);
    break;
    case 4: display(); break;
  }
 } while(ch!=0);
}
void append(int num)
{
  struct dnode *nptr, *temp = start;
  nptr = malloc(sizeof(struct dnode));
  nptr -> data = num;
  nptr -> next = NULL;
  nptr -> prev = NULL;
  if(start == NULL)
```

```
    {
      start = nptr; }
    else
      {
        while(temp -> next != NULL)
        temp = temp -> next;
        nptr -> ptr = temp;
        temp -> next = nptr;
      }
  }
void insert (int num)
{
  struct dnode (int num)
  { nptr = malloc (sizeof (struct dnode));
    nptr -> prev = NULL;
   nptr -> data = num;
   nptr -> next = start;
   if (start != NULL)
   start -> prev = nptr;
   start = nptr;
  }
 void remove (int num)
{ struct dnode * temp = start;
  while (temp != NULL)
  { if(temp -> data == num)
    { if (temp == start)
      { start = start -> next;
        start -> prev = NULL;
      }
```

DATE: __/__/____
PAGE _____

Experiment- 5

Aim: Create circular list having information about colleges & perform insertion at front & del. at end.

Software Used: Turbo C.

Source Code

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct node
{
   int colcode;
   char colname[30];
   struct node *next;
};
struct node *ptr1, *ptr2, *start = NULL;
void create(int n)
{
   int i;
   for(i = 0; i < n; i++)
   {
      ptr1 = start -> next;
      if(start == NULL)
      {
         ptr1 = (struct node *)malloc(sizeof(struct node));
         printf("Enter college code & its name: \n");
```

GOOD WRITE

DATE: __/__/____
PAGE _____

```
    scanf("%d %s", &ptr1 -> colcode, &ptr1 -> colname);
    start = ptr1;
    start -> next = NULL;
  }

else if (ptr1 == NULL)
{
  ptr1 = (struct node*) malloc (sizeof (struct node));
  printf("Enter college code & its name: \n");
  scanf(" %d %s", &ptr1 -> colcode, &ptr1 -> colname);
  start -> next = ptr1;
  ptr1 -> next = start;
  }
else
    { for ptr1 = start -> next; ptr1 -> next != start;
        ptr1 = ptr1 -> next);
      ptr2 = (struct node*) malloc(sizeof(struct node));
      printf("Enter college code & its name: \n");
      scanf("%d %s", &ptr2 -> collcode, &ptr2 -> colname);
      ptr1 -> next = ptr2;
      ptr2 -> next = start;
    }
}}
Void insertbeg()
{
  for (ptr2 = start -> next; ptr2 -> next) = start; ptr2 = ptr2 -> next
  ptr1 = (struct node*) malloc (sizeof(struct node));
  printf("Enter college code & its name: ");
```

GOOD WRITE

```
else{
        if(temp -> next == NULL)
        temp -> prev -> next = NULL;
    else
        {
          temp -> prev -> next = temp -> next;
          temp -> next -> prev = temp -> prev;
        }
     free(temp);
    }
    return;
}
  temp = temp -> next;
   }
  printf("\n%.d not found !!", num);
}
void display()
{
  struct dnode * temp = start;
  printf("\n");
  while(temp != NULL)
  {
    printf("%.d\t", temp -> data);
    temp = temp -> next;
}
```

9/10/18

DATE: __/__/____
PAGE _____

```
    scanf("%d%s", &ptr1->collcode, &ptr1->colname);
    ptr1->next = start;
    start = ptr1;
    ptr2->next = ptr1;
}
void delend()
{
  if(start == NULL)
      printf("The list is empty!!");
  else if(start->next == NULL)
      free(start);
  else
      {
          for(ptr1 = start->next; ptr1->next != start;
              ptr1 = ptr1->next);
          ptr2 = ptr1;
          ptr2->next = start;
          free(ptr1);
}}
void printlist()
```

GOOD WRITE

DATE: __/__/____
PAGE _____

```
{ if (start == NULL)
  printf ("The circular list is empty! \n");
  else {
        printf("The college code is: %d \n", start -> colcode);
        printf("The college name is: %d \n", start -> colname);
        if (start -> next != NULL)
        {
for(ptr1 = start -> next; ptr1 != start; ptr1 = ptr1 -> next)
  {
   printf("The college code is: %d \n", ptr1 -> colcode);
   printf("The college name is: %d \n", ptr1 -> cdname);
  }
}}}
void main()
{ int n, c;
  char ans;
  clrscr();
  print f("Enter no. of entries for circular list \n");
  scanf("%d, &n);
  create (n);
  do { printf("1: Insert at beginning");
       printf("2: Del in end \n 3: Print circular list");
       scanf ("%d", &c); switch (c) {
case 1 : insert beg(); break; case 2: ddlend(); break;
case 3: printlist; break; default: printf("Invalid choice!");}
printf("Do you want to continue \n");
fflush (stdin); scanf ("%c", &ans); }
while (ans == 'y' || ans == 'Y');
getch(); }
```

GOOD WRITE

Telco Dt.: Pg.:

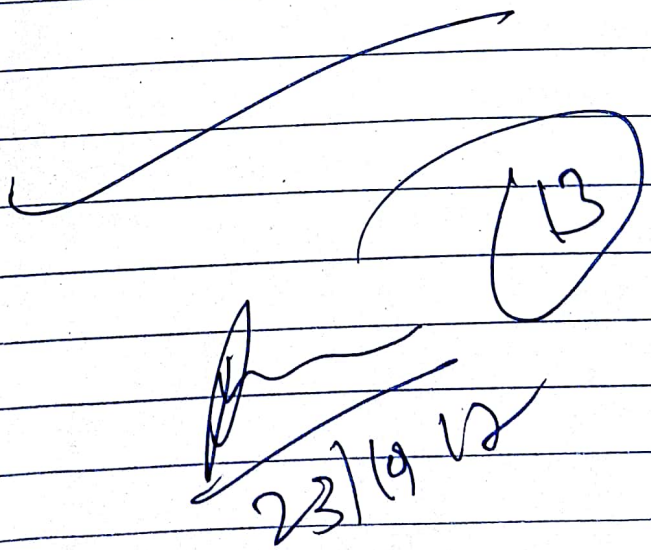
Experiment 6

- **Aim**: Create a stack & perform push, pop & traverse using linear linked list.
- **Software Used**: Turbo C
- **Source Code**

```
#include <stdio.h>
#include <conio.h>
struct node
{
  int data;
  struct Node *next;
} *top = NULL;
void push (int);
void pop ();
void display ();
void main ()
{
 int choice, value;
 clrscr();
 while(1){
        printf("\n1. Push 2. Pop 3. Display 4. Exit");
        printf ("\nEnter your choice:");
        scanf ("%d", &choice);
        switch (choice){
```

```
        case 1: printf("Enter the value to insert:");
                scanf("%d", &value);
                push(value);
                break;
        case 2: pop(); break;
        case 3: display(); break;
        case 4: exit(0);
        default: printf("\n Incorrect option !!");
      }
    }
}
void push(int value)
{
  struct Node *newNode;
  newNode = (struct Node*) malloc(sizeof(struct Node));
  newNode -> data = value;
  if (top == NULL)
      newNode -> Next = NULL;
  else
      newNode -> Next = top;
      top = newNode;
      printf("\n Inserted successfully!");
}
void pop()
{
  if (top == NULL)
      printf("\n Stack is Empty !!!!!");
```

```
else
    struct Node * temp = top;
    printf("\n Deleted element: %d", temp->data);
    top = temp->next;
    free(temp);
  }
}
void display()
{
 if(top == NULL)
   printf("\n Stack is Empty !!!!!");
 else{
     struct Node * temp = top;
     while(temp != NULL){
           printf("%d\n" temp -> data);
           temp = temp -> next;
          }
    }
}
```



DATE: __/__/____
PAGE _____

Experiment 7

Aim : Create a linear queue using array & implement diff. operations like Insert, delete & display the elements.

Software Used: Turbo C 7

Source Code

```
#define MAX 50
#include <stdio.h>
int queue array[MAX];
int rear = -1;
int front = -1;
int main()
{
  int choice;
  while (1){
        printf("1. Insert element \n 2. Delete Element \n
                3. Display \n 4. Quit \n Enter choice:");
        scanf("%d", &ch);
        switch(ch)
        {
        case 1 : insert(); break;
        case 2 : delete(); break;
        case 3 : display(); break;
        default : printf("Incorrect choice!");
```

```
insert ()
{
  int add;
  if (rear == MAX - 1)
      printf ("Overflow !!");
  else { if
       (front == -1)
            front = 0;
  printf ("Insert the element in queue :");
  Scanf (" %d", &add);
  rear = rear + 1;
  queue_array [rear] = add;
} }
delete ()
{ ;if (front == -1 || front == rear)
  { printf ("Queue Underflow !! "); return;
    else
        { printf (" Element deleted from queue is %d \n ",
          queue_array [front]); front = front + 1 ); } }
}

display ()
{
  int i;
  if ( front == -1)
  printf (" Queue is empty ");
  else{
       printf("Queue is :"); for(int i=front; i<= rear;
                                 i++);
```

}}{ printf("%d", queue_array[i]); }

Output

1. Insert Element
2. Delete Element
3. Display
4. Quit

Enter choice: 1

Insert the element in queue: 50

1. Insert element
2. Delete element
3. Display
4. Quit

Enter choice: 3

50

Telco Dt.: Pg.:

Experiment 8

- **Aim**: To create a binary tree & perform traversal operations — Inorder, Preorder & Postorder.
- **Software Used**: Turbo C7

- **Source Code**

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct node
{
  int data;
  struct node *right, *left;
} *root, *p, *q;

struct node *make(int y)
{
  struct node *newnode;
  newnode = (struct node *)malloc(sizeof(struct node));
  newnode -> data = y;
  newnode -> right = newnode -> left = NULL;
  return(newnode);
}
void left(struct node *r, int x)
{
  if(r -> left != NULL)
    printf("\n Invalid!");
```

```
        else
            r -> left = make(x);
    }
void right(struct node *r, int x)
{
    if(r -> right != NULL)
        printf("\n Invalid!");
    else
        r -> right = make(x);
    }

void inorder(struct node *r)
{
    if(r != NULL)
    {
        inorder(r -> left);
        printf("%d\t", r -> data);
        inorder(r -> right);
    }
}
void preorder(struct node *r)
{
    if(r != NULL)
    {
        printf("%d\t", r -> data);
        preorder(r -> left);
        preorder(r -> right);
    }
}
```

```
void postorder (struct node *r)
{
  if(r != NULL)
  {
    postorder (r -> left);
    postorder (r -> right);
    printf("%d\t", r -> data);
  }
}
void create ()
{
  int no;
  char choice = 'y';
  printf ("\nEnter the root : ");
  scanf("%d", &no);
  root = make(no);
  p = root;
  while (choice == 'y' || choice == 'Y')
  {
    printf ("\nEnter number : ");
    scanf("%d", &no);
    if (no == -1)
        break;
    p = root;
    q = root;
    while (no != p -> data && q != NULL)
    {
```

```
        p = q;
        if (no < p -> data)
                q = p -> left;
        else
                q = p -> right;
    }
    if (no < p -> data)
    {
        printf("Left branch of %d is %d \n",
                                p -> data, no);
        left (p, no);
    }
    else{
        right (p, no);
        printf("-Right branch of %d is %d \n",
                                p -> data, no);
        }
    printf("Continue? (y/n):");
    scanf ("%c", &choice);
    }
}
void main()
{
  int no, action;
  clrscr();
  while (1)
  {
```

```
printf("\n\n1. Create \n");
printf("2. Inorder Traversal \n");
printf("3. Preorder Traversal \n");
printf("4. Postorder Traversal \n");
printf("5. Exit.\n");
printf("Enter choice: ");
scanf("%d", &action)
switch(action)
{
  case 1: create; break;
  case 2: printf("\n\nInorder Traversal is: \n");
          inorder(root);
          break;
  case 3: printf("\n\nPreorder Traversal is: \n")
          preorder(root);
          break;
  case 4: printf("\n\nPostorder Traversal is: \n")
          postorder(root);
          break;
  default:
          exit(0);
          break;
  }
getch();
}
}
```