LABORATORY MANUAL
FOR
DATA STRUCTURE USING C

3RD Semester

Diploma in Computer Science & Engineering

Department of Computer Science & Engineering

C. V. Raman Polytechnic

Bidya Nagar, Mahura, Janla, Bhubaneswar

(Affiliated to SCTE&VT: Approved by AICTE)

Experiment 1:
Implementation of 1D & 2D Array

Experiment 1.1
Write a program to traverse an array using C.

Source Code:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[7], i=0;
clrscr();
printf("Enter the elements of the array :");
for (i=0;i<=6 ;i++ )
{
   scanf("%d\n", &a[i]);
   }
printf("\nThe elements of the array you have enetered are as follows :")
for(i=0; i<=6; i++)
   {
  printf("%d", a[i]);
}
getch();
}
```

OUTPUT:-
Enter the elements of the array : 7
The elements of the array you have entered are as follows :
12
23
34
45
56
67
78

Experiment 1.2

Write a program to insert an element to an array using C.

Source Code:

```
#include <stdio.h>
void main()
{
   int array[100];
   int i, n, x, pos;
   printf("Enter the number of elements in the array \n");
   scanf("%d", &n);
   printf("Enter the elements \n");
   for (i = 0; i < n; i++)
   {
      scanf("%d", &array[i]);
   }
   printf("Input array elements are: \n");
   for (i = 0; i < n; i++)
   {
      printf("%d ", array[i]);
   }
   printf("\nEnter the new element to be inserted: ");
   scanf("%d", &x);
   printf("Enter the position where element is to be inserted: ");
   scanf("%d", &pos);
   n=n+1;
   for(i = n-1; i >= pos; i--)
      array[i]=array[i-1];

   array[pos-1]=x;
   for (i = 0; i < n; i++)
   {
      printf("%d ", array[i]);
   }
}
```

OUTPUT:-

```
Enter the number of elements in the array
5
Enter the elements
2 7 1 23 5
Input array elements are:
2 7 1 23 5
Enter the new element to be inserted: 15
Enter the position where element is to be inserted: 4
2 7 1 15 23 5
```

Experiment 1.3

Write a program to delete an element to an array using C.

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   int i, n, index, arr[10];
   printf("Enter the size of the array: ");
   scanf("%d", &n);
   printf("Enter the elements of the array: \n");
   for (i = 0; i < n; i++)
   {
      printf("arr[%d] = ", i);
      scanf("%d", &arr[i]);
   }
   printf("Enter the index of the element to be deleted: ");
   scanf("%d", &index);
   if (index >= n+1)
   {
      printf (" \n Deletion is not possible in the array.");
   }
   else
   {
      for (i = index; i < n - 1; i++)
         arr[i] = arr[i + 1];
         printf("The array after deleting the element is: ");
      for (i = 0; i < n - 1; i++)
         printf("%d ", arr[i]);
      return 0;
   }
}
```

OUTPUT:-

```
Enter the size of the array: 5
Enter the elements of the array:
arr[0] = 8
arr[1] = 4
arr[2] = 9
arr[3] = 6
arr[4] = 2
Enter the index of the element to be deleted: 3
The array after deleting the element is: 8 4 9 2
```

Experiment 1.3

Write a program to sum two matrix by implementing 2D array using C.

Source Code:

```
#include <stdio.h>
int main() {
  int r, c, a[100][100], b[100][100], sum[100][100], i, j;
  printf("Enter the number of rows (between 1 and 100): ");
  scanf("%d", &r);
  printf("Enter the number of columns (between 1 and 100): ");
  scanf("%d", &c);
  printf("\nEnter elements of 1st matrix:\n");
  for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
      printf("Enter element a%d%d: ", i + 1, j + 1);
      scanf("%d", &a[i][j]);
    }
  printf("Enter elements of 2nd matrix:\n");
  for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
      printf("Enter element b%d%d: ", i + 1, j + 1);
      scanf("%d", &b[i][j]);
    }
  for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
      sum[i][j] = a[i][j] + b[i][j];
    }
  printf("\nSum of two matrices: \n");
  for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
      printf("%d   ", sum[i][j]);
      if (j == c - 1) {
        printf("\n\n");
      }
    }
  return 0;
}
```

OUTPUT:-

```
Enter the number of rows (between 1 and 100): 2
Enter the number of columns (between 1 and 100): 3

Enter elements of 1st matrix:
Enter element a11: 2
Enter element a12: 3
Enter element a13: 4
Enter element a21: 5
```

Enter element a22: 2
Enter element a23: 3
Enter elements of 2nd matrix:
Enter element b11: -4
Enter element b12: 5
Enter element b13: 3
Enter element b21: 5
Enter element b22: 6
Enter element b23: 3

Sum of two matrices:
-2 8 7

10 8 6

Experiment 2:-

Write a program to implement stack using C.

Source Code:

```
#include<stdio.h>
int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()
{
   top=-1;
   printf("\n Enter the size of STACK[MAX=100]:");
   scanf("%d",&n);
   printf("\n\t STACK OPERATIONS USING ARRAY");
   printf("\n\t--------------------------------");
   printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
   do
   {
      printf("\n Enter the Choice:");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
         {
            push();
            break;
         }
         case 2:
         {
            pop();
            break;
         }
         case 3:
         {
            display();
            break;
         }
         case 4:
         {
            printf("\n\t EXIT POINT ");
            break;
         }
         default:
         {
            printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
         }
```

```
        }
    }
    while(choice!=4);
    return 0;
}
void push()
{
    if(top>=n-1)
    {
        printf("\n\tSTACK is over flow");
    }
    else
    {
        printf(" Enter a value to be pushed:");
        scanf("%d",&x);
        top++;
        stack[top]=x;
    }
}
void pop()
{
    if(top<=-1)
    {
        printf("\n\t Stack is under flow");
    }
    else
    {
        printf("\n\t The popped elements is %d",stack[top]);
        top--;
    }
}
void display()
{
    if(top>=0)
    {
        printf("\n The elements in STACK \n");
        for(i=top; i>=0; i--)
            printf("\n%d",stack[i]);
        printf("\n Press Next Choice");
    }
    else
    {
        printf("\n The STACK is empty");
    }

}
```

OUTPUT:-

```
Enter the size of STACK[MAX=100]:10

        STACK OPERATIONS USING ARRAY
       -----------------------------
        1.PUSH
        2.POP
        3.DISPLAY
        4.EXIT
 Enter the Choice:1
 Enter a value to be pushed:12
 Enter the Choice:1
 Enter a value to be pushed:24
 Enter the Choice:1
 Enter a value to be pushed:98
 Enter the Choice:3
 The elements in STACK
98
24
12
 Press Next Choice
 Enter the Choice:2
        The popped elements is 98
 Enter the Choice:3
 The elements in STACK
24
12
 Press Next Choice
 Enter the Choice:4

        EXIT POINT
```

Experiment 3:
Write a program to implement pointer using C.

What is a pointer in C?
A pointer is defined as a derived data type that can store the address of other C variables or a memory location. We can access and manipulate the data stored in that memory location using pointers.

Application of Pointer
i) It is used to access array elements
ii) It is used for dynamic memory allocation.
iii) It is used in Call by reference
iv) It is used in data structures like trees, graph, linked list etc.

Source Code:
```
#include <stdio.h>
void pointerDemo()
{
int var1 = 30;
int *ptr1;
int **ptr2;
ptr1 = &var1;
ptr2 = &ptr1;
printf("Value at ptr1 = %p \n",ptr1);
printf("Value at var1 = %d \n",var1);
printf("Value of variable using *ptr1 = %d \n", *ptr1);
printf("Value at ptr2 = %p \n",ptr2);
printf("Value stored at *ptr2 = %d \n", *ptr2);
printf("Value of variable using  **ptr2 = %d \n", **ptr2);
}
int main()
{
pointerDemo();
return 0;
}
```

OUTPUT:-
```
Value at ptr1 = 0x7fffc57e1c5c
Value at var1 = 30
Value of variable using *ptr1 = 30
Value at ptr2 = 0x7fffc57e1c60
Value stored at *ptr2=-981590948
Value of variable using **ptr2 = 30
```

Experiment 4:
Structure & Union

Experiment 4.1
Write a program to implement Structure using C.

Source Code:
```
#include<stdio.h>
typedef struct
{
   int id;
   char name[36];
   int sal;
} EMP;
EMP getdata ()
{
   EMP te;
   printf ("Enter EMP ID:");
   scanf ("%d", &te.id);
   printf ("Enter EMP name:");
   scanf ("%s", te.name);
   printf ("Enter EMP salary:");
   scanf ("%d", &te.sal);
   return te;
}
void showdata (EMP te)
{
   printf ("\nID : %d NAME:%s SALARY:%d", te.id, te.name, te.sal);
}
int sumsal (int s1, int s2)
{
   return (s1 + s2);
}
int main ()
{
   EMP e1, e2;
   int tsal;
   e1 = getdata ();
   e2 = getdata ();
   showdata (e1);
   showdata (e2);
   tsal = sumsal (e1.sal, e2.sal);
   printf ("\nSum Salary = %d", tsal);
}
```

OUTPUT:-

```
Enter EMP ID:101
Enter EMP name: Manisha
Enter EMP salary:20000
Enter EMP ID:102
Enter EMP name: Kalpana
Enter EMP salary: 30000
ID: 101 NAME: Manisha
SALARY:20000
ID: 102 NAME: Kalpana
SALARY: 30000
Sum Salary = 50000
```

Experiment 4.2

Write a program to implement Union using C.

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{

   int i;
   struct employee_imfo
   {
      int employee_id;
      union
      {
         long int adhar_card_number;
         long int voter_id_card_number;
         char other_id [8];
      };
      char goverment_id;
   }a,b,c;

   a.employee_id = 8;
   a.goverment_id='a';
   a.adhar_card_number = 868796;

   b.employee_id=7;
   b.goverment_id='v';
   b.voter_id_card_number = 1234;

   printf("Employee a Information :\n");
   if(a.goverment_id=='a')
   {
      printf("GOVT. ID PROVIDED IS ADHAR CARD_ID\n");
   }
   else
   {
      printf("GOVT. ID PROVIDED IS VOTER ID CARD\n");
   }

   printf("a id :%d\na's government provided id %d", a.employee_id,a.adhar_card_number);
   printf("\n\n");

   if(b.goverment_id=='a')
   {
      printf("GOVT. ID PROVIDED IS ADHAR CARD_ID\n");
   }
```

```
    else
    {
        printf("GOVT. ID PROVIDED IS VOTER ID CARD\n");
    }

    printf("Employee b Information :\n");
    printf("b id :%d\nb's government provided id %d",
b.employee_id,b.voter_id_card_number);
    printf("\n\n");

    printf("size of structure variable a %d", sizeof(a));
    return 0;

}
```

OUTPUT:-

```
Employee a Information:
GOVT. ID PROVIDED IS ADHAR CARD_ID a id: 8
a's government provided id 868796

GOVT. ID PROVIDED IS VOTER ID CARD
Employee b Information: b id: 7
b's government provided id 1234

size of structure variable a  12
```

Experiment 5:-

Write a program to implement push and pop operation on Stack using C.

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4
int top = -1, inp_array[SIZE];
void push();
void pop();
void show();

int main()
{
   int choice;

   while (1)
   {
      printf("\nPerform operations on the stack:");
      printf("\n1.Push the element\n2.Pop the element\n3.Show\n4.End");
      printf("\n\nEnter the choice: ");
      scanf("%d", &choice);

      switch (choice)
      {
      case 1:
         push();
         break;
      case 2:
         pop();
         break;
      case 3:
         show();
         break;
      case 4:
         exit(0);

      default:
         printf("\nInvalid choice!!");
      }
   }
}

void push()
{
   int x;

   if (top == SIZE - 1)
```

```
    {
        printf("\nOverflow!!");
    }
    else
    {
        printf("\nEnter the element to be added onto the stack: ");
        scanf("%d", &x);
        top = top + 1;
        inp_array[top] = x;
    }
}

void pop()
{
    if (top == -1)
    {
        printf("\nUnderflow!!");
    }
    else
    {
        printf("\nPopped element: %d", inp_array[top]);
        top = top - 1;
    }
}

void show()
{
    if (top == -1)
    {
        printf("\nUnderflow!!");
    }
    else
    {
        printf("\nElements present in the stack: \n");
        for (int i = top; i >= 0; --i)
            printf("%d\n", inp_array[i]);
    }
}
```

OUTPUT:-

Perform operations on the stack:
1.Push the element
2.Pop the element
3.Show
4.End

Enter the choice: 1

Enter the element to be inserted onto the stack: 10

Perform operations on the stack:
1.Push the element
2.Pop the element
3.Show
4.End

Enter the choice: 3

Elements present in the stack:
10

Perform operations on the stack:
1.Push the element
2.Pop the element
3.Show
4.End

Enter the choice: 2
Popped element: 10

Perform operations on the stack:
1.Push the element
2.Pop the element
3.Show
4.End

Enter the choice: 3
Underflow!!

Experiment: 6

Write a program to implement insertion and deletion in Queue.

Source Code:

```
#include <stdio.h>
# define SIZE 100
void enqueue();
void dequeue();
void show();
int inp_arr[SIZE];
int Rear = - 1;
int Front = - 1;
main()
{
   int ch;
   while (1)
   {
      printf("1.Enqueue Operation\n");
      printf("2.Dequeue Operation\n");
      printf("3.Display the Queue\n");
      printf("4.Exit\n");
      printf("Enter your choice of operations : ");
      scanf("%d", &ch);
      switch (ch)
      {
         case 1:
         enqueue();
         break;
         case 2:
         dequeue();
         break;
         case 3:
         show();
         break;
         case 4:
         exit(0);
         default:
         printf("Incorrect choice \n");
      }
   }
}

void enqueue()
{
   int insert_item;
   if (Rear == SIZE - 1)
      printf("Overflow \n");
   else
```

```
    {
        if (Front == - 1)

        Front = 0;
        printf("Element to be inserted in the Queue\n : ");
        scanf("%d", &insert_item);
        Rear = Rear + 1;
        inp_arr[Rear] = insert_item;
    }
}

void dequeue()
{
    if (Front == - 1 || Front > Rear)
    {
        printf("Underflow \n");
        return ;
    }
    else
    {
        printf("Element deleted from the Queue: %d\n", inp_arr[Front]);
        Front = Front + 1;
    }
}

void show()
{

    if (Front == - 1)
        printf("Empty Queue \n");
    else
    {
        printf("Queue: \n");
        for (int i = Front; i <= Rear; i++)
            printf("%d ", inp_arr[i]);
        printf("\n");
    }
}
```

OUTPUT:-

```
1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 1
Element to be inserted in the Queue: 10

1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
```

4.Exit
Enter your choice of operations : 1
Element to be inserted in the Queue: 20

1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 3
Queue:
10 20

1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 2
Element deleted from the Queue: 10

1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations: 3
Queue:
20

Experiment 7:-

Write a program to implement insertion and deletion in Linked List using C.

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
void create();
void display();
void insert_begin();
void insert_end();
void insert_pos();
void delete_begin();
void delete_end();
void delete_pos();
struct node* head = NULL;
struct node
{
        int data;
        struct node* next;
};
int main()
{
        int choice;
        while(1)
        {
           printf("\n*****\n");
           printf("0. Create\n");
           printf("1. display\n");
           printf("2. Insert Node at beginning\n");
           printf("3. Insert Node in specific position\n");
           printf("4. Insert Node at end of LinkedList\n");
           printf("5. Delete Node at beginning\n");
           printf("6. Delete Node at end\n");
           printf("7. Delete Node at position\n");
           printf("8. ** To exit **");

                printf("\n Enter your choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 0: create();
                                        break;
                        case 1: display();
                                        break;
                        case 2: insert_begin();
                                        break;
                        case 3: insert_pos();
                                        break;
```

```
                        case 4: insert_end();
                                break;
                        case 5: delete_begin();
                                break;
                        case 6: delete_end();
                                break;
                        case 7: delete_pos();
                                break;
                        case 8: exit(0);
                        default:printf("\n Wrong Choice");
                break;
                    }
        }
}
void create()
{
        struct node* temp;
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d",&temp->data);
        temp->next = NULL;
        if(head==NULL)      {
                head = temp;
        }
        else{
                struct node* ptr = head;
                while(ptr->next!=NULL)
                {
                        ptr = ptr->next;
                }
                ptr->next = temp; //inserting at end of List
        }
}
void display()
{
        if(head==NULL)
        {
                printf("Linked List is Empty\n");
                return;
        }
      printf("LinkedList: ");
        struct node* ptr = head;
        while(ptr!=NULL) // start from first node
        {
                printf("%d ",ptr->data);
                ptr = ptr->next;
        }
   printf("\n");
}
void insert_begin()
```

```
{
        struct node* temp;
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d",&temp->data);
        temp->next = NULL;
        if(head==NULL)
        {
                head = temp;
                return;
        }
        else
        {
                temp->next = head; //point it to old head node
                head = temp; //point head to new first node
        }
}
void insert_pos()
{
        struct node* temp;
      // creating a new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d",&temp->data);
        temp->next = NULL;
        if(head==NULL) // if list empty we return
        {
                head = temp;
             return;
        }
        else
        {
                struct node* prev_ptr;
                struct node* ptr = head;
                int pos;
                printf("Enter position: ");
                scanf("%d",&pos);
                for(int i=0;i<pos;i++)
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                temp->next = ptr;
                prev_ptr->next = temp;
        }
}
void insert_end()
{
        struct node* temp;
        temp = (struct node*)malloc(sizeof(struct node));
```

```
        printf("Enter node data: ");
        scanf("%d",&temp->data);
        temp->next = NULL;
        if(head==NULL)
        {
                head = temp; //if list is empty, we return
              return;
        }
        else{
                struct node* ptr = head;
                while(ptr->next!=NULL)
                {
                        ptr = ptr->next;
                }
                        ptr->next = temp;
        }
}
void delete_begin()
{
        if(head==NULL) //if List is empty we return
        {
                printf("Linked List is empty | Nothing to delete \n");
                return;
        }
        else
        {
                struct node* ptr = head;
                head = head->next; // head node pointing to second node
                free(ptr); // deleting prev head node
                printf("Node Deleted \n");
        }
}

void delete_end()
{
        if(head==NULL) //if List is empty we return
        {
                printf("Linked List is empty | Nothing to delete \n");
                return;
        }
        else if(head->next==NULL)
        {
                struct node* ptr = head;
                head = ptr->next;
                free(ptr);
        }
        else
        {
                struct node* ptr = head;
                struct node* prev_ptr = NULL;
```

```
                while(ptr->next!=NULL)// traverse till last but one node
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                prev_ptr->next = NULL; // next field of last but one field is made as NULL
                free(ptr); // deleting last node
        }
}
void delete_pos()
{
        int pos;
        printf("Enter node position to delete: ");
        scanf("%d",&pos);
        struct node* ptr=head;
        if(head==NULL) //we return if List is empty
        {
                printf("Linked List is empty \n");
                return;
        }
        else if(pos == 0)
        {
                ptr = head;
                head=ptr->next; // head pointing to second node
                free(ptr); // deleting old first node
        }
        else
        {
                struct node* prev_ptr;
                for(int i=0;i<pos;i++)
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                prev_ptr->next = ptr->next; //prev node pointing to pos+1 node
                free(ptr); //deleting node at pos
        }
}
```

OUTPUT:-

0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position
8. ** To exit **

```
 Enter your choice: 0
Enter node data: 12
*****

0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position
8. ** To exit **
 Enter your choice: 1
LinkedList: 12
*****

0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position
8. ** To exit **
 Enter your choice: 2
Enter node data: 13
*****

0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position
8. ** To exit **
 Enter your choice: 3
Enter node data: 25
Enter position: 1

*****
0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
```

6. Delete Node at end
7. Delete Node at position
8. ** To exit **
Enter your choice: 4
Enter node data: 39
*****

0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position
8. ** To exit **
Enter your choice: 5
Node Deleted
*****

0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position
8. ** To exit **
Enter your choice: 6
Last Node Deleted
*****

0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position
8. ** To exit **
Enter your choice: 7
Enter node position to delete: 2
Node at pos: 2 deleted
*****

Experiment 8:-

Write a program to implement Bubble sort using C.

Source Code:

```
#include <stdio.h>
void bubblesort(int arr[], int size)
{
   int i, j;
   for (i = 0;  i < size; i++)
   {
      for (j = 0; j < size - i; j++)
      {
         if (arr[j] > arr[j+1])
            swap(&arr[j], &arr[j+1]);
      }
   }
}
void swap(int *a, int *b)
{
   int temp;
   temp = *a;
   *a = *b;
   *b = temp;
}
int main()
{
   int array[100], i, size;
   printf("How many numbers you want to sort:  ");
   scanf("%d", &size);
   printf("\nEnter %d numbers : ", size);
   for (i = 0; i < size; i++)
      scanf("%d", &array[i]);
   bubblesort(array, size);
   printf("\nSorted array is ");

   for (i = 0; i < size; i++)
      printf(" %d ", array[i]);
   return 0;
}
```

OUTPUT:-

```
How many numbers you want to sort: 5
Enter 5 numbers : 345 3 20 35 333
Sorted array is : 3 20 35 333 345
```

Experiment 9:-

Write a program to implement Quick sort using C.

Source Code:

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

int quickSort(int *arr, int low, int high)
{
   int i = low, j = high;
   int pivot = arr[(low + high) / 2];
   while (i <= j)
   {
      while (arr[i] < pivot)
         i++;
      while (arr[j] > pivot)
         j--;
      if (i <= j)
      {
         int temp = arr[i];
         arr[i] = arr[j];
         arr[j] = temp;
         i++;
         j--;
      }
   }
   if (low < j)
      quickSort(arr, low, j);
   if (i < high)
      quickSort(arr, i, high);
   return 0;
}

int main(void)
{
   puts("Enter the number of elements in the array: ");
   int n;
   scanf("%d", &n);
   int arr[n];
   puts("Enter the elements of the array: ");
   for (int i = 0; i < n; i++)
   {
      printf("arr[%d]: ", i);
      scanf("%d", &arr[i]);
   }
   int low = 0;
   int high = n - 1;
   int pivot = arr[high];
```

```
    int k = low - 1;
    for (int j = low; j < high; j++)
    {
        if (arr[j] <= pivot)
        {
            k++;
            int temp = arr[k];
            arr[k] = arr[j];
            arr[j] = temp;
        }
    }
    int temp = arr[k + 1];
    arr[k + 1] = arr[high];
    arr[high] = temp;
    int pi = k + 1;
    quickSort(arr, low, pi - 1);
    quickSort(arr, pi + 1, high);
    puts("The sorted array is: ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

OUTPUT:-

```
Enter the number of elements in the array:
8
Enter the elements of the array:
arr[0]: 4
arr[1]: 7
arr[2]: 2
arr[3]: 9
arr[4]: 11
arr[5]: 3
arr[6]: 1
arr[7]: 6
The sorted array is:
1 2 3 4 6 7 9 11
```

Experiment 10:-

Write a program to implement Binary tree traversal using C.

Source Code:

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

/* A binary tree node has data, pointer to left child
 and a pointer to right child */
struct node {
    int data;
    struct node* left;
    struct node* right;
};

/* Helper function that allocates a new node with the
 given data and NULL left and right pointers. */
struct node* newNode(int data) {
    struct node* node = (struct node*) malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return (node);
}

/* Given a binary tree, print its nodes according to the
 "bottom-up" postorder traversal. */
void printPostorder(struct node* node) {
    if (node == NULL)
        return;

    // first recur on left subtree
    printPostorder(node->left);

    // then recur on right subtree
    printPostorder(node->right);

    // now deal with the node
    printf("%d ", node->data);
}

/* Given a binary tree, print its nodes in inorder*/
void printInorder(struct node* node) {
    if (node == NULL)
        return;

    /* first recur on left child */
```

```
    printInorder(node->left);
    printf("%d ", node->data);

    printInorder(node->right);
}

void printPreorder(struct node* node) {
    if (node == NULL)
        return;
    printf("%d ", node->data);
    printPreorder(node->left);
    printPreorder(node->right);
}
int main() {
    struct node *root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);

    printf("\n Preorder traversal of binary tree is \n");
    printPreorder(root);

    printf("\n Inorder traversal of binary tree is \n");
    printInorder(root);

    printf("\n Postorder traversal of binary tree is \n");
    printPostorder(root);

    getchar();
    return 0;
}
```

OUTPUT:-

```
Preorder traversal of binary tree is
1 2 4 5 3
 Inorder traversal of binary tree is
4 2 5 1 3
 Postorder traversal of binary tree is
4 5 2 3 1
```

Experiment:-11

Write a program to implement Linear Search using C.

Source Code:

```
#include <stdio.h>
void main()
{
 int num;
    int i,  keynum, found = 0;
    printf("Enter the number of elements ");
    scanf("%d", &num);
    int array[num];
    printf("Enter the elements one by one \n");
    for (i = 0; i < num; i++)
    {
        scanf("%d", &array[i]);
    }
    printf("Enter the element to be searched ");
    scanf("%d", &keynum);
    for (i = 0; i < num ; i++)
    {
        if (keynum == array[i] )
        {
            found = 1;
            break;
        }
    }
    if (found == 1)
        printf("Element is present in the array at position %d",i+1);
    else
        printf("Element is not present in the array\n");
}
```

OUTPUT:-

```
Enter the number of elements 6
Enter the elements one by one
4
6
1
2
5
3
Enter the element to be searched 6
Element is present in the array at position 2
```

Experiment 12:-

Write a program to implement Binary Search using C.

Source Code:

```
#include <stdio.h>
int binarySearch(int a[], int beg, int end, int val)
{
   int mid;
   if(end >= beg)
   {       mid = (beg + end)/2;
      if(a[mid] == val)
      {
         return mid+1;
      }
      else if(a[mid] < val)
      {
         return binarySearch(a, mid+1, end, val);
      }
      else
      {
         return binarySearch(a, beg, mid-1, val);
      }
   }
   return -1;
}
int main() {
  int a[] = {11, 14, 25, 30, 40, 41, 52, 57, 70}; // given array
  int val = 40; // value to be searched
  int n = sizeof(a) / sizeof(a[0]); // size of array
  int res = binarySearch(a, 0, n-1, val); // Store result
  printf("The elements of the array are - ");
  for (int i = 0; i < n; i++)
  printf("%d ", a[i]);
  printf("\nElement to be searched is - %d", val);
  if (res == -1)
  printf("\nElement is not present in the array");
  else
  printf("\nElement is present at %d position of array", res);
  return 0;
}
```

OUTPUT:-

```
The elements of the array are - 11 14 25 30 40 41 52 57 70
Element to be searched is - 40
Element is present at 5 position of array
```