PRACTICAL FILE
ON
DATA SCIENCE STRUCTURE USING C [CSIT124]

SUBMITTED TO:
MS NEETU NARAYAN

SUBMITTED BY:
Name: Nandini Sain
Enrolment No.: A2305221060
B.Tech. 3CSE6X

COMPUTER SCIENCE AND ENGINEERING (CSE)
AMITY SCHOOL OF ENGINEERING &TECHNOLOGY
AMITY UNIVERSITY, UTTAR PRADESH
SESSION- 2022-2023

NAME- NANDINI SAIN

ENROLLMENT NO. A2305221060

3CSE6(X)

DATA STRUCTURES IN C LAB REPORT

INDEX

S.no	Program	Submitted on	Signature
1.	Insertion of Linear Array	18/08/2022	
2.	Traversal of a Linear Array	18/08/2022	
3.	Deletion of a Linear Array	18/08/2022	
4.	Linear Search in an Array	18/08/2022	
5.	Binary Search in an Array	18/08/2022	
6.	Bubble Sort in an Array	15/09/2022	
7.	Search Second Largest Value in an Array	15/09/2022	
8.	Calculate String Length (without strlen function)	15/09/2022	
9.	Stack Operations (To implement Insertion, Deletion Operation in a stack)	6/10/2022	
10.	Tower of Hanoi	6/10/2022	
11.	Queue Operation (To implement Insertion, Deletion Operation in a queue)	6/10/2022	
12.	Circular Queue Operations (To implement Insertion, Deletion Operation in a circular queue)	6/10/2022	
13.	Creation of a Linked List	6/10/2022	
14.	Linked List Operations (To implement Insertion, Deletion Operation in a linked list)	6/10/2022	
15.	Circular Linked List Operations (To implement Insertion, Deletion Operation in a circular linked list)	3/11/2022	
16.	Stack Implementation Operations (To perform operations on stack implemented using Linked list	3/11/2022	
17.	Creation of Tree Traversal	3/11/2022	
18.	Binary Search Tree Operations	10/11/2022	
19.	To perform Insertion Sort	10/11/2022	
20.	To perform Selection Sort	10/11/2022	
21.	To perform Heap Sort	10/11/2022	
22.	To perform Merge Sort	10/11/2022	
23.	To perform Quick Sort	10/11/2022	
24.	To perform Shell Sort	10/11/2022	

Data Science using C Lab Work

1. Insertion of Linear Array

```
#include <stdio.h>
int main()
{
   int A[10] = {2,4,6,8,9};
  int x = 7, k=3, n = 5;
  int i = 0, j = n;

  printf("original array elements:\n");

  for(i = 0; i<n; i++)
  {
    printf("A[%d] = %d \n", i, A[i]);
  }
  n = n + 1;

  while( j >= k)
  {
    A[j+1] = A[j];
    j = j - 1;
  }

  A[k] = x;
  printf("array elements after insertion:\n");

  for(i = 0; i<n; i++)
  {
    printf("A[%d] = %d \n", i, A[i]);
  }
}
```

OUTPUT

```
original array elements:
A[0] = 2
A[1] = 4
A[2] = 6
A[3] = 8
A[4] = 9
array elements after insertion:
A[0] = 2
A[1] = 4
A[2] = 6
A[3] = 7
A[4] = 8
A[5] = 9
```

2. Traversal of a Linear Array

```
#include <stdio.h>
int main()
{
   int A[] = {2,4,6,8,9};
  int i, n = 5;

  printf("The array elements are:\n");

  for(i = 0; i < n; i++)
  {
     printf("A[%d] = %d \n", i, A[i]);
  }
}
```

Output

```
The array elements are:
A[0] = 2
A[1] = 4
A[2] = 6
A[3] = 8
A[4] = 9
```

3. Deletion of a Linear Array

```
#include <stdio.h>
int main()
{
  int A[10] = {2,4,6,8,9};
  int k = 4, n = 5;
  int i, j;

  printf("Original array elements:\n");

  for(i = 0; i<n; i++)
  {
    printf("A[%d] = %d \n", i, A[i]);
  }

  j = k;

  while( j < n)
  {
   A[j-1] = A[j];
    j = j + 1;
  }

  n = n -1;

  printf("Array elements after deletion:\n");

  for(i = 0; i<n; i++)
  {
    printf("A[%d] = %d \n", i, A[i]);
  }
}
```

Output

```
Original array elements:
A[0] = 2
A[1] = 4
A[2] = 6
A[3] = 8
A[4] = 9
Array elements after deletion:
A[0] = 2
A[1] = 4
A[2] = 6
A[3] = 9
```

4. Linear Search in an Array

```
#include <stdio.h>
int main ()
{
   int a[7], n, i, x;
   printf("\nEnter size of Array ");
   scanf("%d", &n);
   printf("\nEnter Elements\n");
   for(i=0; i<n; i++)
   {
      scanf("%d",&a[i]);
   }
   printf("Enter elements to be searched");
   scanf("%d", &x);
   for(i=0; i<n;i++)
   {
      if(a[i]==x)
      {
         printf("\n Element is at position %d", i+1);
      }
   }
   return 0;
}
```

OUTPUT

```
Enter size of Array
Enter Elements
4
5
6
```

7

8

Enter elements to be searched 5

Element is at position 2

5. Binary Search in an Array

```
#include <stdio.h>
int main()
{
  int c, first, last, middle, n, search, array[100];
  printf("Enter number of elements\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);
  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);

  printf("Enter value to find\n");
  scanf("%d", &search);

  first = 0;
  last = n - 1;
  middle = (first+last)/2;

  while (first <= last) {
    if (array[middle] < search)
      first = middle + 1;
    else if (array[middle] == search) {
      printf("%d found at location %d.\n", search, middle+1);
      break;
    }
  else
    last = middle - 1;
  middle = (first + last)/2;
}
```

```
  if (first > last)
    printf("Not found! %d isn't present in the list.\n", search);

  return 0;
}
```

OUTPUT

```
Enter number of elements
4
Enter 4 integers
5
6
7
8
Enter value to find
6
6 found at location 2.
```

6. Bubble Sort in an Array

```
#include<stdio.h>
int main()
{
int a[10],i,j,temp;
printf("\nEnter the elements of the array:");
for(i=0;i<10;i++)
{
        scanf("%d",&a[i]);
}
for(i=0;i<10;i++)
{
        for(j=0;j<10-i-1;j++)
        {
                if(a[j]>a[j+1])
                {
                        temp=a[j];
                        a[j]=a[j+1];
                        a[j+1]=temp;
                }
        }
}
printf("\nThe sorted array is:");
for(i=0;i<10;i++)
{
        printf("%d\t",a[i]);
}
return 0;
}
```

OUTPUT

Enter the elements of the array:7

8

9

1

2

3

4

5

6

7

The sorted array is:1 2 3 4 5 6 7 7 8 9

7. Search Second Largest Value in an Array

```
#include <stdio.h>
int main()
{
int a[10],i,j,n,a1,a2;

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

a1 = a[0],a2 = a[n-1];

if(a1<a2)
{
a1=a2;
a2=a[0];
}

for(i=0;i<n;i++)
{
for(j=i+1;j<n-i;j++)
{
if(a[j]>a1)
{
a1 = a[j];
}
```

```
else if(a2< a[j]< a1)
{
a2 = a[j];
}
}
}

printf("Second largest number is :%d\n",a2);
return 0;
}
```

OUTPUT

```
Enter the size of the array:
5
Enter the array
1
3
5
7
9
Second largest number is :7
```

8. Calculate String Length (without strlen function)

```
#include <stdio.h>


int main()
{
   char Arr[100];
   int length=0;
   printf ("Enter string: ");
   scanf("%s",Arr);


   for (int i=0;Arr[i] != '\0';i++)
   {
      length++;
   }
   printf ("Length of string is: %d",length);


   return 0;
}
```

OUTPUT

```
Enter string: Apple
Length of string is: 5
```

9. Stack Operations (To implement Insertion, Deletion Operation in a stack)

```
include<stdio.h>
int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()
{
   //clrscr();
   top=-1;
   printf("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("\nEnter the Choice: ");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
         {
            push();
            break;
         }
         case 2:
         {
            pop();
            break;
         }
         case 3:
         {
            display();
            break;
         }
         case 4:
         {
            printf("\nEXIT POINT ");
            break;
         }
         default:
         {
            printf ("\n\tPlease Enter a Valid Choice(1/2/3/4)\n");
         }
```

```
        }
    }
    while(choice!=4);
    return 0;
}
void push()
{
    if(top>=n-1)
    {
        printf("STACK is over flow\n");

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

}
```

OUTPUT

```
Enter the size of STACK[MAX=100]: 3
        STACK OPERATIONS USING ARRAY
       --------------------------------
        1.PUSH
        2.POP
        3.DISPLAY
        4.EXIT
Enter the Choice: 1
Enter a value to be pushed:5

Enter the Choice: 1
Enter a value to be pushed:6

Enter the Choice: 3
The elements in STACK:
6
5
Press Next Choice

Enter the Choice: 2
The popped element is: 6
Enter the Choice: 2
The popped element is: 5
Enter the Choice: 2
Stack is under flow

Enter the Choice: 4
EXIT POINT
```

10. Tower of Hanoi

```
#include <stdio.h>

void TOH(int n, char A, char B, char C)
{
   if (n==1)
   {
      printf("\nMove disk 1 from rod %c to rod %c",A ,C);
      return;
   }
   TOH(n-1,A,C,B);
   printf("\nMove disk %d from rod %c to rod %c",n, A, C);
   TOH(n-1, B, A, C);
}

int main()
{
   int n;
   printf("Enter no. of disks: ");
   scanf("%d",&n);

   TOH(n,'A','B','C');

   return 0;
}
```

OUTPUT

```
Enter no. of disks: 3

Move disk 1 from rod A to rod C
Move disk 2 from rod A to rod B
Move disk 1 from rod C to rod B
Move disk 3 from rod A to rod C
Move disk 1 from rod B to rod A
Move disk 2 from rod B to rod C
Move disk 1 from rod A to rod C
```

11. Queue Operation (To implement Insertion, Deletion Operation in a queue)

```
#include <stdio.h>

int main()
{
   int A[10];
   int rear=-1;
   int front=-1;
   int choice=0;
   while (choice < 4)
   {
      printf("\nEnter choice for Push(1), Pop(2), Display(3), Exit(4): ");
      scanf("%d",&choice);
      switch (choice)
      {
         case 1: if (front == 0 && rear == 9)
               {
                  printf("Queue overflow\n");
                  break;
               }
               else
               {
                  int element;
                  printf("Enter element to insert: ");
                  scanf("%d",&element);
                  front = 0;
                  rear++;
                  A[rear]=element;
               }
               break;

         case 2: if (rear == -1 && front == -1)
               {
                  printf("Queue underflow\n");
                  break;
               }
               else
               {
                  front++;
               }
               break;

         case 3: if (rear == -1 && front == -1)
               {
```

```
            printf("No elements\n");
            break;
        }
        else
        {
            for (int i=front; i<=rear; i++)
            {
                printf("%d\n",A[i]);
            }
        }
        break;

    case 4: break;
    default: printf("\nInvalid choice\n");


    }
  }
  return 0;
}
```

OUTPUT

```
Enter choice for Push(1), Pop(2), Display(3), Exit(4): 1
Enter element to insert: 3

Enter choice for Push(1), Pop(2), Display(3), Exit(4): 1
Enter element to insert: 5

Enter choice for Push(1), Pop(2), Display(3), Exit(4): 3
3
5

Enter choice for Push(1), Pop(2), Display(3), Exit(4): 2

Enter choice for Push(1), Pop(2), Display(3), Exit(4): 2

Enter choice for Push(1), Pop(2), Display(3), Exit(4): 4
```

12. Circular Queue Operations (To implement Insertion, Deletion Operation in a circular queue)

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5
int queue[MAX];
int front=-1;
int rear=-1;
void insert();
void delete();
void display();
int main()
{
   int choice;
   while(1)
   {
      printf("\n1.Insert\n2.Delete\n3.Display\n4.Exit\n");
      printf("Enter your choice:");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1: insert();
               break;
         case 2: delete();
               break;
         case 3: display();
               break;
         case 4: exit(0);

         default: printf("Wrong choice");
      }
   }
}
void insert()
{
   int item;
   if((front==0 && rear==MAX-1) || (front==rear+1))
   {
      printf("Queue Overflow");
      return;
   }
   printf("Enter the element to be inserted:");
   scanf("%d",&item);
   if(rear==-1)
   {
      front=0;
```

```
      rear=0;
   }
   else if(rear==MAX-1 && front!=0)
   {
      rear=0;
   }
   else
   {
      rear=rear+1;
   }
   queue[rear]=item;
}
void delete()
{
   if(front==-1)
   {
      printf("Queue Underflow");
      return;
   }
   printf("Element deleted from queue is:%d",queue[front]);
   if(front==rear)
   {
      front=-1;
      rear=-1;
   }
   else if(front==MAX-1)
   {
      front=0;
   }
   else
   {
      front=front+1;
   }
}
void display()
{
   int i;
   if(front==-1)
   {
      printf("Queue is empty");
      return;
   }
   printf("Queue elements are: ");
   if(front<=rear)
   {
      for(i=front;i<=rear;i++)
      {
         printf("%d ",queue[i]);
```

```
    }
  }
  else
  {
    for(i=front;i<=MAX-1;i++)
    {
      printf("%d ",queue[i]);
    }
    for(i=0;i<=rear;i++)
    {
      printf("%d ",queue[i]);
    }
  }
  printf(" ");
}
```

OUTPUT

```
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1
Enter the element to be inserted: 5

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1
Enter the element to be inserted: 8

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 3
Queue elements are: 5 8
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 2
Element deleted from queue is: 5
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 4
```

13. Creation of a Linked List

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *head=NULL;
struct node *current=NULL;
void insert(int data)
{
   struct node *link=(struct node*)malloc(sizeof(struct node));
   link->data=data;
   link->next=NULL;
   if(head==NULL)
   {
      head=link;
      current=link;
   }
   else
   {
      current->next=link;
      current=link;
   }
}
void display()
{
   struct node *ptr=head;
   while(ptr!=NULL)
   {
      printf("%d\t",ptr->data);
      ptr=ptr->next;
   }
}
int main()
{
   int n;
   printf("Enter the number of elements:");
   scanf("%d",&n);
   for(int i=0;i<n;i++)
   {
      int data;
      printf("Enter the element:");
      scanf("%d",&data);
      insert(data);
   }
```

```
    display();
    return 0;
}
```

OUTPUT

```
Enter the number of elements:5
Enter the element:5
Enter the element:6
Enter the element:7
Enter the element:8
Enter the element:9
5      6      7      8      9
```

14. Linked List Operations (To implement Insertion, Deletion Operation in a linked list)

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *head = NULL;
void insertele(int val)
{
   struct node *newnode, *p;
   newnode = (struct node*)malloc(sizeof(struct node));
   newnode->data = val;
   newnode->next = NULL;
   if(head == NULL)
   {
      head = newnode;
   }
   else
   {
      p = head;
      while(p->next != NULL)
      {
         p = p->next;
      }
      p->next = newnode;
   }
}
void printlist()
{
```

```
    struct node *p = head;
    while (p != NULL)
    {
        printf("%d ",p->data);
        p = p->next;
    }
}
void insertbeg(int val)
{
    struct node *newnode1,*p;
    newnode1 = (struct node*)malloc(sizeof(struct node));
    newnode1->data = val;
    newnode1->next = head;
    head = newnode1;
}
void insertend(int val)
{
    struct node *newnode2, *p;
    newnode2 = (struct node*)malloc(sizeof(struct node));
    newnode2->data = val;
    newnode2->next = NULL;
    p=head;
    while(p->next != NULL)
    {
        p = p->next;
    }
    if (p->next == NULL)
    {
        p->next = newnode2;
        newnode2->next = NULL;
    }
```

```
}

void insertpos(int val, int pos)
{
   struct node *newnode3, *ptr;
   newnode3 = (struct node*)malloc(sizeof(struct node));
   newnode3->data = val;
   newnode3->next = NULL;
   ptr = head;
   for (int i=0; i<pos-2; i++)
   {
      ptr = ptr->next;
   }
   newnode3->next = ptr->next;
   ptr->next = newnode3;
}
void delpos(int pos)
{
   struct node *p;
   p = head;
   if (head == NULL)
   {
      printf("List underflow!");
      return;
   }
   else
   {
      if (p->next == NULL && pos == 1)
      {
         head = NULL;
         return;
      }
```

```
        else
        {
            for (int i=0; i<pos-2; i++)
            {
                p = p->next;
            }
            p->next = p->next->next;
        }
    }
}
void srchelement(int srchele)
{
    struct node *p;
    int found = 0;
    p = head;
    while (p != NULL)
    {
        if (p->data == srchele)
        {
            found = 1;
        }
        p = p->next;
    }
    if (found)
    {
        printf("Element found!");
    }
    else
    {
        printf("Element not found!");
    }
}
```

```
int main()
{
   int n;
   int ele;
   int newele1;
   int newele2;
   int newele3;
   int pos;
   int srchele;
   int choice;
   printf("Enter number of nodes(max 8): ");
   scanf("%d",&n);

   if (n > 8)
   {
      printf("Nodes cannot be created more than 8");
   }
   else
   {
      for (int i=0; i<n ; i++)
      {
         printf("Enter elements: ");
         scanf("%d",&ele);
         insertele(ele);
      }

      while (choice < 7)
      {
         printf("\n\nEnter choice for \nInsert at beginning(1) \nInsertion at end(2) \nInsert at given
position(3) \nDelete any element(4) \nSearch element(5) \nDisplay List(6) \nExit(7) \nEnter choice:
");
         scanf("%d",&choice);
         switch (choice)
```

```
        {
            case 1: printf("Enter element: ");
                    scanf("%d",&newele1);
                    insertbeg(newele1);
                    break;
            case 2: printf("Enter element: ");
                    scanf("%d",&newele2);
                    insertend(newele2);
                    break;
            case 3: printf ("Enter element: ");
                    scanf("%d",&newele3);
                    printf("Enter position: ");
                    scanf("%d",&pos);
                    insertpos(newele3,pos);
                    break;
            case 4: printf("Enter position: ");
                    scanf("%d",&pos);
                    delpos(pos);
                    break;
            case 5: printf("Enter element to search: ");
                    scanf("%d",&srchele);
                    srchelement(srchele);
                    break;
            case 6: printlist();
                    break;
            case 7: break;
            default: printf("Invalid choice!");
        }
      }
   }
   return 0;
}
```

OUTPUT

```
Enter number of nodes(max 8): 5
Enter elements: 4
Enter elements: 5
Enter elements: 6
Enter elements: 7
Enter elements: 8

Enter choice for
Insert at beginning(1)
Insertion at end(2)
Insert at given position(3)
Delete any element(4)
Search element(5)
Display List(6)
Exit(7)
Enter choice: 1
Enter element: 2

Enter choice for
Insert at beginning(1)
Insertion at end(2)
Insert at given position(3)
Delete any element(4)
Search element(5)
Display List(6)
Exit(7)
Enter choice: 2
Enter element: 3

Enter choice for
```

Insert at beginning(1)

Insertion at end(2)

Insert at given position(3)

Delete any element(4)

Search element(5)

Display List(6)

Exit(7)

Enter choice: 3

Enter element: 3

Enter position: 5

Enter choice for

Insert at beginning(1)

Insertion at end(2)

Insert at given position(3)

Delete any element(4)

Search element(5)

Display List(6)

Exit(7)

Enter choice: 4

Enter position: 1

Enter choice for

Insert at beginning(1)

Insertion at end(2)

Insert at given position(3)

Delete any element(4)

Search element(5)

Display List(6)

Exit(7)

Enter choice: 5

Enter element to search: 4

Element not found!

Enter choice for

Insert at beginning(1)

Insertion at end(2)

Insert at given position(3)

Delete any element(4)

Search element(5)

Display List(6)

Exit(7)

Enter choice: 5

Enter element to search: 2

Element found!

Enter choice for

Insert at beginning(1)

Insertion at end(2)

Insert at given position(3)

Delete any element(4)

Search element(5)

Display List(6)

Exit(7)

Enter choice: 6

2 5 6 3 7 8 3

Enter choice for

Insert at beginning(1)

Insertion at end(2)

Insert at given position(3)

Delete any element(4)

Search element(5)

Display List(6)

Exit(7)

Enter choice: 7

15. Circular Linked List Operations (To implement Insertion, Deletion Operation in a circular linked list)

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

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

struct node *head = NULL;

void insertele(int val)
{
   struct node *newnode, *p;
   newnode = (struct node*)malloc(sizeof(struct node));
   newnode->data = val;
   newnode->next = NULL;

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

```
        {
            p = p->next;
        }
        p->next = newnode;
        newnode->next = head;
    }
}

void printlist()
{
    struct node *p = head;
    while (p->next != head)
    {
        printf("%d ",p->data);
        p = p->next;
    }
    printf("%d ",p->data);
}

void insertbeg(int val)
{
    struct node *newnode1,*p;
    newnode1 = (struct node*)malloc(sizeof(struct node));
    newnode1->data = val;
    newnode1->next = head;

    p = head;
    while(p->next != head)
    {
        p = p->next;
    }
```

```
    p->next = newnode1;
    head = newnode1;

}

void insertend(int val)
{
    struct node *newnode2, *p;
    newnode2 = (struct node*)malloc(sizeof(struct node));
    newnode2->data = val;
    newnode2->next = head;

    p = head;
    while(p->next != head)
    {
        p = p->next;
    }
    p->next = newnode2;
}

void deletebeg()
{
    struct node *p;
    p = head;
    while(p->next != head)
    {
        p = p->next;
    }
    p->next = head->next;
    head = head->next;
}
```

```
void deleteend()
{
   struct node *p, *q;
   p = head;
   while(p->next != head)
   {
      q = p;
      p = p->next;
   }
   q->next = head;
}

void deleteele(int val)
{
   struct node *p, *q;
   p = head;
   while(p->data != val)
   {
      q = p;
      p = p->next;
   }
   q->next = p->next;
}

int main()
{
   int ch, val;
   while(1)
   {
```

```
    printf("1. Insert element 2. Insert at beginning 3. Insert at end 4. Delete at beginning 5. Delete at end 6. Delete element 7. Print list 8. Exit");
    printf("\nEnter your choice: ");
    scanf("%d",&ch);
    switch(ch)
    {
      case 1: printf("\nEnter the value to be inserted: ");
              scanf("%d",&val);
              insertele(val);
              break;
      case 2: printf("\nEnter the value to be inserted: ");
              scanf("%d",&val);
              insertbeg(val);
              break;
      case 3: printf("\nEnter the value to be inserted: ");
              scanf("%d",&val);
              insertend(val);
              break;
      case 4: deletebeg();
              break;
      case 5: deleteend();
              break;
      case 6: printf("\nEnter the value to be deleted: ");
              scanf("%d",&val);
              deleteele(val);
              break;
      case 7: printlist();
              break;
      case 8: exit(0);
      default: printf("\nInvalid choice");
    }
```

```
    }
    return 0;
}
```

OUTPUT

```
1. Insert element 2. Insert at beginning 3. Insert at end 4. Delete at beginning 5. Delete at end 6. Delete element 7. Print list 8. Exit

Enter your choice: 1

Enter the value to be inserted: 4

1. Insert element 2. Insert at beginning 3. Insert at end 4. Delete at beginning 5. Delete at end 6. Delete element 7. Print list 8. Exit

Enter your choice: 2

Enter the value to be inserted: 3

1. Insert element 2. Insert at beginning 3. Insert at end 4. Delete at beginning 5. Delete at end 6. Delete element 7. Print list 8. Exit

Enter your choice: 3

Enter the value to be inserted: 2

1. Insert element 2. Insert at beginning 3. Insert at end 4. Delete at beginning 5. Delete at end 6. Delete element 7. Print list 8. Exit

Enter your choice: 7

3 4 2

1. Insert element 2. Insert at beginning 3. Insert at end 4. Delete at beginning 5. Delete at end 6. Delete element 7. Print list 8. Exit

Enter your choice: 4

1. Insert element 2. Insert at beginning 3. Insert at end 4. Delete at beginning 5. Delete at end 6. Delete element 7. Print list 8. Exit

Enter your choice: 3

Enter the value to be inserted: 8

1. Insert element 2. Insert at beginning 3. Insert at end 4. Delete at beginning 5. Delete at end 6. Delete element 7. Print list 8. Exit

Enter your choice: 8
```

16. Stack Implementation Operations (To perform operations on stack implemented using Linked list

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

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

struct node *head = NULL;

void insertele(int val)
{
   struct node *newnode, *p;
   newnode = (struct node*)malloc(sizeof(struct node));
   newnode->data = val;
   newnode->next = NULL;

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

void printlist()
{
   struct node *p = head;
   while (p->next != head)
   {
      printf("%d ",p->data);
      p = p->next;
```

```
    }
    printf("%d ",p->data);
}

void insertbeg(int val)
{
    struct node *newnode1,*p;
    newnode1 = (struct node*)malloc(sizeof(struct node));
    newnode1->data = val;
    newnode1->next = head;

    p = head;
    while(p->next != head)
    {
        p = p->next;
    }
    p->next = newnode1;
    head = newnode1;

}

void insertend(int val)
{
    struct node *newnode2, *p;
    newnode2 = (struct node*)malloc(sizeof(struct node));
    newnode2->data = val;
    newnode2->next = head;

    p = head;
    while(p->next != head)
    {
        p = p->next;
    }
    p->next = newnode2;
}

void deletebeg()
{
    struct node *p;
    p = head;
    while(p->next != head)
    {
        p = p->next;
    }
    p->next = head->next;
    head = head->next;
}
```

```
void deleteend()
{
   struct node *p, *q;
   p = head;
   while(p->next != head)
   {
      q = p;
      p = p->next;
   }
   q->next = head;
}

void deleteele(int val)
{
   struct node *p, *q;
   p = head;
   while(p->data != val)
   {
      q = p;
      p = p->next;
   }
   q->next = p->next;
}

int main()
{
   int ch, val;
   while(1)
   {
      printf("1. Insert element 2. Insert at beginning 3. Insert at end 4. Delete at beginning 5.
Delete at end 6. Delete element 7. Print list 8. Exit");
      printf("\nEnter your choice: ");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1: printf("\nEnter the value to be inserted: ");
               scanf("%d",&val);
               insertele(val);
               break;
         case 2: printf("\nEnter the value to be inserted: ");
               scanf("%d",&val);
               insertbeg(val);
               break;
         case 3: printf("\nEnter the value to be inserted: ");
               scanf("%d",&val);
               insertend(val);
               break;
         case 4: deletebeg();
```

```
            break;
        case 5: deleteend();
            break;
        case 6: printf("\nEnter the value to be deleted: ");
            scanf("%d",&val);
            deleteele(val);
            break;
        case 7: printlist();
            break;
        case 8: exit(0);
        default: printf("\nInvalid choice");
      }
   }
   return 0;
}
```

OUTPUT

```
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1

Enter the value to be inserted: 4

1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1

Enter the value to be inserted: 8

1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1

Enter the value to be inserted: 5

1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 3

5

8

4

1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 2

Deleted element is 5
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 4
```

17.Creation of Tree Traversal

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *left;
struct node *right;
};
struct node *root=NULL;
struct node *insert(struct node *,int);
void inorder(struct node *);
void preorder(struct node *);
void postorder(struct node *);
void main()
{
int choice,n;
do
{
printf("\n1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit");
printf("\nEnter your choice: ");
scanf("%d",&choice);
switch(choice)
{
case 1:
printf("\nEnter the value: ");
scanf("%d",&n);
root=insert(root,n);
break;
case 2:
printf("\nInorder Traversal is: ");
inorder(root);
break;
case 3:
printf("\nPreorder Traversal is: ");
preorder(root);
break;
case 4:
printf("\nPostorder Traversal is: ");
postorder(root);
break;
case 5:
exit(0);
}
}while(1);
```

```
}
struct node *insert(struct node *root,int n)
{
struct node *temp,*cur,*prev;
temp=(struct node *)malloc(sizeof(struct node));
temp->data=n;
temp->left=NULL;
temp->right=NULL;
if(root==NULL)
root=temp;
else
{
cur=root;
while(cur!=NULL)
{
prev=cur;
if(n<cur->data)
cur=cur->left;
else
cur=cur->right;
}
if(n<prev->data)
prev->left=temp;
else
prev->right=temp;
}
return root;
}
void inorder(struct node *root)
{
if(root!=NULL)
{
inorder(root->left);
printf("%d",root->data);
inorder(root->right);
}
}
void preorder(struct node *root)
{
if(root!=NULL)
{
printf("%d",root->data);
preorder(root->left);
preorder(root->right);
}
}
void postorder(struct node *root)
{
```

```
if(root!=NULL)
{
postorder(root->left);
postorder(root->right);
printf("%d",root->data);
}
}
```

OUTPUT

```
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter your choice: 1

Enter the value: 3

1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter your choice: 1

Enter the value: 8

1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter your choice: 1

Enter the value: 9

1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter your choice: 1

Enter the value: 5

1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter your choice: 2

Inorder Traversal is: 3589
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter your choice: 3

Preorder Traversal is: 3859
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter your choice: 4

Postorder Traversal is: 5983
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter your choice: 5
```

18.Binary Search Tree Operations

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *left;
struct node *right;
};
struct node *root=NULL;
struct node *insert(struct node *,int);
struct node *delete(struct node *,int);
void inorder(struct node *);
void preorder(struct node *);
void postorder(struct node *);
void main()
{
int ch,n;
do
{
printf("\n1.Insert 2.Delete 3.Inorder 4.Preorder 5.Postorder 6.Exit\n");
printf("\nEnter your choice: ");
scanf("%d",&ch);
switch(ch)
{
case 1:printf("\nEnter the number to be inserted: ");
scanf("%d",&n);
root=insert(root,n);
break;
case 2:printf("\nEnter the number to be deleted: ");
scanf("%d",&n);
root=delete(root,n);
break;
case 3:inorder(root);
break;
case 4:preorder(root);
break;
case 5:postorder(root);
break;
case 6:exit(0);
}
}while(ch!=6);
}
struct node *insert(struct node *root,int n)
{
if(root==NULL)
```

```
{
root=(struct node *)malloc(sizeof(struct node));
root->data=n;
root->left=NULL;
root->right=NULL;
}
else if(n<root->data)
root->left=insert(root->left,n);
else if(n>root->data)
root->right=insert(root->right,n);
return root;
}
struct node *delete(struct node *root,int n)
{
struct node *temp;
if(root==NULL)
{
printf("\nElement not found");
return root;
}
else if(n<root->data)
root->left=delete(root->left,n);
else if(n>root->data)
root->right=delete(root->right,n);
else
{
if(root->left==NULL)
{
temp=root->right;
free(root);
return temp;
}
else if(root->right==NULL)
{
temp=root->left;
free(root);
return temp;
}
else
{
temp=root->right;
while(temp->left!=NULL)
temp=temp->left;
root->data=temp->data;
root->right=delete(root->right,temp->data);
}
}
return root;
```

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

OUTPUT

```
1.Insert 2.Delete 3.Inorder 4.Preorder 5.Postorder 6.Exit

Enter your choice: 1

Enter the number to be inserted: 3

1.Insert 2.Delete 3.Inorder 4.Preorder 5.Postorder 6.Exit

Enter your choice: 1

Enter the number to be inserted: 5

1.Insert 2.Delete 3.Inorder 4.Preorder 5.Postorder 6.Exit

Enter your choice: 1

Enter the number to be inserted: 9
```

1.Insert 2.Delete 3.Inorder 4.Preorder 5.Postorder 6.Exit

Enter your choice: 1

Enter the number to be inserted: 8

1.Insert 2.Delete 3.Inorder 4.Preorder 5.Postorder 6.Exit

Enter your choice: 1

Enter the number to be inserted: 1

1.Insert 2.Delete 3.Inorder 4.Preorder 5.Postorder 6.Exit

13589
1.Insert 2.Delete 3.Inorder 4.Preorder 5.Postorder 6.Exit

Enter your choice: 4
31598
1.Insert 2.Delete 3.Inorder 4.Preorder 5.Postorder 6.Exit

Enter your choice: 5
18953
1.Insert 2.Delete 3.Inorder 4.Preorder 5.Postorder 6.Exit

Enter your choice: 6

19.To perform Insertion Sort

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
int a[10],i,j,n,temp;
printf("Enter the number of elements: ");
scanf("%d",&n);
printf("Enter the elements: ");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=1;i<n;i++)
{
temp=a[i];
j=i-1;
while((temp<a[j])&&(j>=0))
{
a[j+1]=a[j];
j=j-1;
}
a[j+1]=temp;
}
printf("The sorted array is: ");
for(i=0;i<n;i++)
printf("%d ",a[i]);
}
```

OUTPUT

```
Enter the number of elements: 4
Enter the elements: 6
2
1
7
The sorted array is: 1 2 6 7
```

20.To perform Selection Sort

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
int a[10],i,j,n,temp,min;
printf("Enter the number of elements: ");
scanf("%d",&n);
printf("Enter the elements: ");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<n-1;i++)
{
min=i;
for(j=i+1;j<n;j++)
{
if(a[j]<a[min])
min=j;
}
temp=a[i];
a[i]=a[min];
a[min]=temp;
}
printf("The sorted array is: ");
for(i=0;i<n;i++)
printf("%d ",a[i]);
}
```

OUTPUT

```
Enter the number of elements: 5
Enter the elements: 6
2
1
7
9
The sorted array is: 1 2 6 7 9
```

21. To perform Heap Sort

```
#include<stdio.h>
#include<stdlib.h>
void heapify(int a[],int n,int i)
{
int largest=i;
int l=2*i+1;
int r=2*i+2;
if((l<n)&&(a[l]>a[largest]))
largest=l;
if((r<n)&&(a[r]>a[largest]))
largest=r;
if(largest!=i)
{
int temp=a[i];
a[i]=a[largest];
a[largest]=temp;
heapify(a,n,largest);
}
}
void heapsort(int a[],int n)
{
int i;
for(i=n/2-1;i>=0;i--)
heapify(a,n,i);
for(i=n-1;i>=0;i--)
{
int temp=a[0];
a[0]=a[i];
a[i]=temp;
heapify(a,i,0);
}
}
void main()
{
int a[10],i,n;
printf("Enter the number of elements: ");
scanf("%d",&n);
printf("Enter the elements: ");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
heapsort(a,n);
printf("The sorted array is: ");
for(i=0;i<n;i++)
printf("%d ",a[i]);
}
```

OUTPUT

```
Enter the number of elements: 5
Enter the elements: 6
2
1
7
4
The sorted array is: 1 2 4 6 7
```

22. To perform Merge Sort

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 100
void mergesort(int a[],int low,int high);
void merge(int a[],int low,int mid,int high);
int main()
{
int a[MAX],n,i;
printf("Enter the number of elements in the array");
scanf("%d",&n);
printf("Enter the elements of the array");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
mergesort(a,0,n-1);
printf("The sorted array is");
for(i=0;i<n;i++)
printf("%d",a[i]);
return 0;
}
void mergesort(int a[],int low,int high)
{
int mid;
if(low<high)
{
mid=(low+high)/2;
mergesort(a,low,mid);
mergesort(a,mid+1,high);
merge(a,low,mid,high);
}
}
void merge(int a[],int low,int mid,int high)
{
int i,j,k,c[MAX];
i=low;
j=mid+1;
k=low;
while(i<=mid && j<=high)
{
if(a[i]<a[j])
{
c[k]=a[i];
i++;
}
else
```

```
{
c[k]=a[j];
j++;
}
k++;
}
while(i<=mid)
{
c[k]=a[i];
i++;
k++;
}
while(j<=high)
{
c[k]=a[j];
j++;
k++;
}
for(k=low;k<=high;k++)
a[k]=c[k];
}
```

OUTPUT

```
Enter the number of elements in the array: 6
Enter the elements of the array: 3
8
1
4
9
5
The sorted array is: 1 3 4 5 8 9
```

23. To perform Quick Sort

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 100
void quicksort(int a[],int low,int high);
int partition(int a[],int low,int high);
void swap(int *a,int *b);
int main()
{
int a[MAX],n,i;
printf("Enter the number of elements in the array");
scanf("%d",&n);
printf("Enter the elements of the array");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
quicksort(a,0,n-1);
printf("The sorted array is");
for(i=0;i<n;i++)
printf("%d",a[i]);
return 0;
}
void quicksort(int a[],int low,int high)
{
int j;
if(low<high)
{
j=partition(a,low,high);
quicksort(a,low,j-1);
quicksort(a,j+1,high);
}
}
int partition(int a[],int low,int high)
{
int pivot,i,j;
pivot=a[low];
i=low+1;
j=high;
while(i<=j)
{
while(a[i]<=pivot && i<=high)
i++;
while(a[j]>pivot && j>=low)
j--;
if(i<j)
swap(&a[i],&a[j]);
```

```
}
swap(&a[low],&a[j]);
return j;
}
void swap(int *a,int *b)
{
int temp;
temp=*a;
*a=*b;
*b=temp;
}
```

OUTPUT

```
Enter the number of elements in the array: 4
Enter the elements of the array: 5
2
1
7
The sorted array is: 1 2 5 7
```

24. To perform Shell Sort

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 100
void shellsort(int a[],int n);
int main()
{
int a[MAX],n,i;
printf("Enter the number of elements in the array");
scanf("%d",&n);
printf("Enter the elements of the array");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
shellsort(a,n);
printf("The sorted array is");
for(i=0;i<n;i++)
printf("%d",a[i]);
return 0;
}
void shellsort(int a[],int n)
{
int i,j,k,temp;
for(i=n/2;i>0;i=i/2)
{
for(j=i;j<n;j++)
{
for(k=j-i;k>=0;k=k-i)
{
if(a[k+i]>=a[k])
break;
else
{
temp=a[k];
a[k]=a[k+i];
a[k+i]=temp;
}
}
}
}
}
```

OUTPUT

Enter the number of elements in the array; 6
Enter the elements of the array: 4
9
2
7
1
6
The sorted array is124679