Date :_____

Experiment-1

1) WAP to implement array operation:-
   i) Find avg. of 10 nos using our array
2) Display pattern:-

```
*
# #
* * *
# # # #
```

3) Find the 1st repeating element in an array.
4) find the largest and smallest number.
5) Squaring the odd position element.

1)

```
#include <stdio.h>
 int main(){
int arr[10];
int sum=0;
float avg;
printf("Enter 10 array elements");
for(int i=0;i<10;i++)
{
  scanf("%d",&arr[i]);
}
  for(int i=0; i<10; i++)
{
sum=sum+arr[i];
```

```
    }
    avg = avg/10;
    printf("The average of all elements: %d",
    avg); return 0;
    }
```

2)
```
#include<stdio.h>
int main()
{
for (int i=1; i<=5; i++)
{
for (int j=1; j<=i; j++)
{
if (i%2==0)
{
printf("#");
}
else
{
printf("\n");
}
return 0;
}
```

3)
```
#include<stdio.h>
int main() {
int arr[5]={10,4,56,3,89};
int i;
int big=arr[0];
int small=arr[0];
for(i=1; i<5; i++){
if (arr[i]>big)
big=arr[i];
if(arr[i]<small)
small=arr[i];
}
printf("Greatest no=%d\n", big);
printf("Smallest no=%d\n",small);
return 0;
}
```

Date :

4)
```
#include <stdio.h>
int main(){
int arr[10];
int largest = arr[0];
int smallest = arr[0];
printf("Enter array elements");
for(int i=0; i<10; i++)
{
 scanf("%d",&arr[i]);
}
for(int i=0; i<10; i++)
{
 if(largest<arr[i])
 {
  largest=arr[i];
 }
 if(smallest>arr[i])
 {
  smallest=arr[i];
 }
}
printf("The largest element: %d",largest);
printf("The smallest element: %d",smallest);
}
```

Date :

```
return 0;
}
```

5)
```
#include <stdio.h>
int main(){
int a[10];
printf("Enter 10 elements of an array\n");
for(int i=0; i<10; i++)
{
 scanf("%d",&a[i]);
}
printf("The array after squaring the odd
 position elements are:\n");
for(int i=0; i<10; i++)
{
 if(i%2==0)
 printf("%d\n",a[i]);
 else
 printf("%d\n",(a[i]*a[i]));
}
 return 0;
}
```

→ Programm:-

1) Display the following figures

a)
```
*
**
***
```

b)
```
1
2 2
3 3 3
4 4 4 4
```

c)
```
1
1 2
1 2 3
1 2 3 4
```

d)
```
*
# #
$ $ $
? ? ? ?
```

e)
```
* * * *
 * * *
  * *
   *
```

f)
```
1
 1 2
  3 3 3
 1 2 3 4
 5 5 5 5
```

g)
```
*
* #
* # $
* # $ ?
```

h)
```
*
* *
* * *
* * * *
```

a)
```
#include<stdio.h>
int main(){
int n,i,j;
int r=3;
for(i=1;i<=r;i++){
for(j=1;j<=i;j++){
printf("*");
}
printf("\n");
}
return 0;
}
```

b)
```
#include <stdio.h>
int main (){
int i,j,r=4;
for (i=1;i<=4;i++)
{
for(j=1;j<=i;j++)
{
printf("%d",i);
}
printf("\n");
}
```

```
    return 0;
}
```

c)
```
#include <stdio.h>
int main (){
int i,j,r=4;
for (i=1; i<=4; i++)
{
 for (j=1; j<=i; j++)
 {
 printf ("%d",j);
 }
 printf("\n");
}
 return 0;
}
```

d)
```
#include<stdio.h>
 int main (){
 int i,j;
 for (i=1 ;i<=4; i++)
 {
 for (j=1; j<=i; j++);
 {
```

```
 if (i==1)
 printf("*");
 else if (i==2)
 printf("#")
 else if (i==3)
 printf ("$")
  else if (i==4)
  printf("?");
 }
  printf("\n");
 }
 return 0;
}
```

e)
```
#include<stdio.h>
 int main (){
 int i,j;
 for (j=4;j>=i;j--)
 {
  for (j=4;j>=i;j--)
 {
  printf(" ");
 }
 for (int k=1; k<=i;k++)
```

```
    {
    printf("*");
    }
    printf("\n");
    }
    return 0;
    }
```

f)
```
#include<stdio.h>
int main(){
{
for(int i=1; i<=5; i++)
{
for(int j=1; j<=i; j++)
{
if(j%2==0)
printf("%d", j);
else
printf("%d", i);
}
printf("\n");
}
return 0;
}
```

g)
```
#include<stdio.h>
int main(){
char str[] = "*#$?";
int i,j;
for(i=0; str[i] != '\0'; i++){
for(int j=0; j<=i; j++){
printf("%c", str[j]);
}
printf("\n");
}
return 0;
}
```

h)
```
#include<stdio.h>
int main(){
int i,j;
int rows = 4;
for(i=1; i<=rows; i++){
for(j=1; j<=i; j++){
printf("*");
}
printf("\n");
}
return 0;
}
```

Date :

Experiment-2

1) Search data using linear search-
Consider the following list to perform linear search. (56, 36, 89, 57, 01, 00, 67, 59)

i) Search the item 01 from the above list & write the item is found or not with procedure.

ii) Search the item 55 from the above list & write the item is found or not with procedure.

2) Search data using binary search. (Example).

3) Compare linear search & binary search.

4) State limitations of linear search in terms of time complexity.

Date :

i)
```
#include <stdio.h>
int main(){
 int k;
 int a[] = {56, 36, 89, 57, 01, 00, 67, 59};
 printf("Enter the element to be found \n");
 scanf("%d", &k);
 int c=0;
 int size = (sizeof(a)/sizeof(a[0]));
 for(int i=0; i<size; i++)
  {
  if(k==a[i])
  {
  printf("Element found at position %d", i);
  c++;
 break;
 }
 }
 if(c==0)
 printf("Element not found \n");
 return 0;}
```

```
#include <stdio.h>
int main()
{
int A[100], n, m, l, h, j
printf("Enter number of Array
      elements:");
scanf("%d", n);
printf("Enter key to be searched:");
scanf("%d", k);
printf("\nEnter sorted Arrays
      elements:\n");

for(i=0; i<n; i++)
{
scanf("%d", a[i]);
}
l=0; h=(n-1);
while(n)=1)
{
n=(int)(l+n);
if(n==a(n])
{
printf("\n element found out at
  index: %d" m)
break;
}
else
if(n>a[m])
{
l=m+1;
}
if(n< a(m);
{
h=m-1;
}
}
}
if(n<1)
printf("\n Element not found;

return 0;
}
```

Date :

Experiment - 2

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

{
  int i,j,n;
  int temp=0;
printf("Enter number of array
    elements:");
scanf ("%d",&n);

int arr[n];
printf("Enter array elements:\n");

for(i=0;i<n;i++)

{
   printf("Enter element %d:\n",i);
scanf("%d", arr[i]);
}
for (i=0; i<n; i++)
{
for(i=0; i<n; i++)
{
```

Date :

```
for (j=0; j<n-1; j++)
{
   if(arr[j] > arr[j+1])
   {
     temp = arr[j];

arr[j] = arr[j+1];
  arr[j+1] = temp;
}
}
}
printf("Array sorted in ascending
  order:\n");

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

  return 0;

}
```

```
#include <stdio.h>
int main()
{
int i,j,n,temp=0;
printf("Enter number of array elements:\n");
scanf("%d", &n);
int arr(n);

printf("Enter array elements:\n");
scanf("%d", &arr[i]);
}
for (i=0; i<n; i++)
{
iff (a(i) <a(i))
{
  if (a[i]<a[i])
  {
    r=a[i];
    a=[i]=a[i];
    a=[i]=r,
  }
}
}
```

```
printf("\n Sorted Array, Element in
   desending order:\n");

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

Experiment - 4

insection sort

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

{ int a[100],n,i,j,k,n=1
printf("Enter no of element:");
scanf("%d",&n);
point f("/n Enter elements of
array:");
for(i=0;i<n,i++);
scanf"%d",& a[i]);
for(i=1,i<A,i++);
{
for(i=0,i<i;i++)
{
if(a[i] <a[i])
{
r=a[i]
for(i<=1-1;ik)=i;k-
{
a(k+1) a[b];
{
a[i]=k;
}
```

Date :

```
print f("/n Array, after parsing %d");

for(i=0,ii<n;ii++)
{
print f("%d",a[i]);
}
++pi
}
printf("/n/n sated array in ascendi
elements:);
for(i=0,i<n;++)
{
print f("/n-%d",a(i));
{
return 0;
}
```

Element-5

→
```
#include <stdio.h>
#include <stdio1ab.h>
struct di
int di;
stuct node*n;
};
struct node *h=Null:
void :b(int x)
{
  struct node));
t→ d->x;

t→n= null;

if (!b)
{
   h=t;
}
else
{ struct node* p=h,
  while(p→h)p=p→n;
    p→n=t;
}
```

Date :

```
}
void del(int x)
{
  struct node *p=h,*q=Nub);
while(p&& p→d !=x)
{
   q=p;
   p=p→n;
}
pt(!p)return;
 pt(!q)
    h=h→n;
  else
q→n=p→n;
 free (p);
}
void s(int x)
{
  struct node * p=h, * q=Null
  {
if (p→d==x)
{
 printf("Found");
  return;
```

```
}
p=p->n;
}
printf("Not found");
}
void dis()
{
struct node* p=h;
while(p)
{
printf("%d", p->d);
p=p->n;
}
printf("\n");
}
int main()
{
int c,x;
for(;;)
{
printf("1.IB 2.IE 3.
Search 5.Display 6.Exit \n");
scanf("%d",&c);
if(c==1)
```

```
break;
switch(c)
{
case1
scanf("%d",&x);
scanf(ib(x)
break;

Case 2
scanf("%d",&x);
ie(x);
break;

Case 3
scanf("%d",&x);
del(x);
break;

Case 4
scanf("%d",&x);
s(x);
break;
```

```
case 5:
dis();
break;
}
}
}
```

Date :

Experiment-07

```
#include <stdio.h>
#include <stdlib.h>
#define size 5
void push();
void pop();
void display();
int stack[size];
int top=-1;
int main()
{
int ch;
printf("1.PUSH\n 2.POP\n
3. DISPLAY\n 4.EXIT");
printf("Enter choice:");
scanf("%d", &ch);
switch(ch)
{
```

```
case 1:
push ();
break;

case 2:
pop ();
break;

case 3:
display();
break;

case 4:
exit(0);
break;
}
}
void pop()
{
if (top==-1)
printf("stack is empty");
else
{
printf("Deleted element is %d",stack[top]);
top--;

print("Element are:")
for (int i=0;i<=top;i++)
{
printf("%d ",stack[i]);
}
print("\n");
}
}
```

Experiment - 9

```
#include <stdio.h>
#define size 5
int q[SIZE];
int f=-1, r=-1;

void insert()
{
   int v;
if (r==size-1)
{
 printf("Overflow");
   return;
}
printf("Enter value");
scanf("%d", &v);
if (f==-1)
   f=0
   r++;
   (r)=v;
   printf("\nInserted -> %d", v);
}
void delete()
}
```

```
if (f==-1 || f>r)
  {
   printf ("Underflow");
      return;
  }
case 1
insert();
break;

Case 2:
 delete();
  break;

Case 3:
 display()
 break;
 Case 4:
    return 0;
  }
 }
}
```

Experiment - 10

```
#include <stdio.h>
#define N 5
int q[N], f = -1, r = -1;
void i() {

int x;
if((f==0 && r==N-1) || (r+1)%N==f)
{ printf("Overflow");
  return; }
scanf("%d", &x);
if (f==-1)
  f=r=0;
else
  r=(r+1)%N;
q[r]=x;
printf("Inserted %d", x);
}
void d()
if (f==-1)
{
  printf("Underflow", q[f]);
if (f==r)
  f=r=-1;
else
  f=(f+1)%N;
}
void dis() {
if (f==-1) {
printf("Empty");
  return; }
int i=f;
while(1) {
printf("%d", q[i]);
if (i==r)
  break;
i=(i+1)%N; }
printf("\n"); }
int main

case 1: i();
  break
case 2: d();
  break;
case 3: dis();
  break;
case 4:
  return 0;
}
```

Experiment-11

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int d;
struct node *l,*r;};
struct node *n= malloc (size of(
struct node));
n->d=v;
n->l=n->r=null;
return n;}

if(r==NULL)
    return nom N(v)

if(v->r->d)
  r->l=ins(r->l,v);

else if (v) r->r=ins(r->l,v);

return r;
```

```
int main (){
struct node* r=NULL;
int ch,v;
while (1){
printf ("1.Ins 2.Pre 3.Exit");
scanf("%d",ch);
case1:
printf ("Enter value");
scanf ("%d",&v);
r= ins(r,v);

Case 2:
printf("preorder:");
    pre(r);

Case 3:
    exit (0);
}
}
}
```

Experiment - 8

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define n 100

char s[n][n];
int t=-1;
void p(char *x)
{
    strcpy(s[++t],x);
}
char *a()
{
    return s[t--];
}
int pr(char c)
{
    if (c=='+'||c=='-')
        return 1;
    if (c=='*'||c=='/')
        return 1;
    if (c=='n')
        return 3
    return 0;
}
int op(char c)
{
    return (c=='+'||c=='-'||c=='*'||
    c=='/'||c=='^');
}
void infix to postfix(char *in, char
*post)
{
    char st[n];
    int top=-1, i, k=0;
    for (i=0; in[i]; i++)
    {
        char c=in[i];
    if (isalnum(c)) post[k++]=c;
    else if (c=='(')
```

```
while (stop! =-1 && st(top)!=
(')
post [k++]=st [top--];
  if (top !=-1)
    top--;
}
else if (op(c))
while top !=-1 && pr (st (top)
)>=pr (c)) post [k++]=st(top)

while (top!=-1)
  post [k++] = st [top--];
  post [k]='\0';
}

void rev (char *x)
{
int i = strlen (x); i
for (i=0; i<1/2; i++)
  {
  char temp = x[i];
    x[i]=x[l-i-1];
```

Date :

```
char in[m], po[m], prf[m];
print ("Enter infix:");
scanf("%s", in);
infix to post (in prf);
printf "post fix = %s\n" po);
printf ("Prefix :%s", prf;
return 0;
}
```

Experiment - 12

```
#include <stdio.h>
#define max 10
int adj [MAX][MAX];
int n;
void create by graph () {
int i,j, edges, origin, destin;

printf ("Enter number of vertices:");
scanf ("%d", &n);
for (i=0; i<n; i++) {
for (j=0; j<n; j++) {
adj [i][j] = 0;
printf ("Enter number of edges:");
scanf ("%d", & edges);

for (i=0; <edges; i++) {
if (origin >= n || destin >= n ||
origin destin) {

printf ("Invalid edge! \n");
} else {
adj [origin] [destin] = 1;
}
void display graph () {
int i,j;
printf ("\n adjacency matrix:\n
for (i=0; i<n; i++)
for (j=0; n<n; j++) {
printf ("%d", adj [i][j]);
}
printf ("\n")

case 1:
create graph ();
break;

case 2:
display by graph ();
break;

case 3:
printf ("taking ... \n");
return 0;
```

```
default:
printf("Invalid choice! Try again");

#include <stdio.h>
#define MAX 10
int main(){
unit graph [MAX][MAX];
int i,j,a,c,dest;

printf("Enter the number of vertices");
scanf("%d",&vertices);
printf("Enter number of edges:");
scanf("%d",&edges);
for(int i=0; i<vertices;j++){
 for(j=0 ,j <vertices i;j++){
  graph[i][j]=0;
 }
}
```

```
printf("Enter the edges source
destiny")
for(i=0;i<edges;i++){
scanf("%d%d",&src,&dest);
 graph(src)(dest)=1;
graph[dest][src]=1;
}

printf("\n Adjacency matrix:\n");
for(i=0;i<vertices;i++){
for(j=0;j<vertices;j++){
 printf("%d",graph[i][j]);
}

 printf("\n");

}

return 0;
}
```

```
    if (temp == NULL) {
        printf("Position out of range!\n");
```

Exp - 6

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

struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};

struct Node* head = NULL;

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

void createList(int n) {
    int data, i;
    struct Node *newNode, *temp;

    if (head != NULL) {
        printf("List already created!\n");
        return;
    }

    for (i = 0; i < n; i++) {
        printf("Enter data for node %d: ", i + 1);
        scanf("%d", &data);
        newNode = createNode(data);

        if (head == NULL) {
            head = newNode;
        } else {
            temp->next = newNode;
            newNode->prev = temp;
        }
        temp = newNode;
    }
}

void displayList() {
    struct Node* temp = head;

    if (head == NULL) {
        printf("List is empty!\n");
        return;
    }

    printf("Doubly Linked List: ");
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

void insertNode(int data, int pos) {
    struct Node* newNode = createNode(data);
    struct Node* temp = head;
    int i;

    if (pos == 1) {
        newNode->next = head;
        if (head != NULL)
            head->prev = newNode;
        head = newNode;
        return;
    }

    for (i = 1; i < pos - 1 && temp != NULL; i++) {
        temp = temp->next;
    }
```

```
    if (temp == NULL) {
        printf("Position out of range!\n");
        free(newNode);
        return;
    }

    newNode->next = temp->next;
    newNode->prev = temp;
    if (temp->next != NULL)
        temp->next->prev = newNode;
    temp->next = newNode;
}


void deleteNode(int pos) {
    struct Node* temp = head;
    int i;

    if (head == NULL) {
        printf("List is empty!\n");
        return;
    }

    if (pos == 1) {
        head = head->next;
        if (head != NULL)
            head->prev = NULL;
        free(temp);
        return;
    }

    for (i = 1; i < pos && temp != NULL; i++) {
        temp = temp->next;
    }

    if (temp == NULL) {
        printf("Position out of range!\n");
        return;
    }

    if (temp->next != NULL)
        temp->next->prev = temp->prev;
    if (temp->prev != NULL)
        temp->prev->next = temp->next;

    free(temp);
}


int main() {
    int choice, n, data, pos;

    while (1) {
        printf("\n--- Doubly Linked List Menu ---\n");
        printf("1. Create List\n");
        printf("2. Display List\n");
        printf("3. Insert Node\n");
        printf("4. Delete Node\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
        case 1:
            if (head == NULL) {
                printf("Enter number of nodes: ");
                scanf("%d", &n);
                createList(n);
            } else {
                printf("List already exists!\n");
            }
            break;

        case 2:
            displayList();
            break;

        case 3:
            printf("Enter data to insert: ");
            scanf("%d", &data);
            printf("Enter position: ");
            scanf("%d", &pos);
            insertNode(data, pos);
```

```
            found = 1;
            break;
        }
        temp = temp->next;
        pos++;
    }

    if (!found)
        printf("Node %d not found in the list.\n", key);
}

// Function to count number of nodes
void countNodes() {
    struct Node* temp = head;
    int count = 0;

    while (temp != NULL) {
        count++;
        temp = temp->next;
    }

    printf("Total number of nodes = %d\n", count);
}

// Function to reverse the doubly linked list
void reverseList() {
    struct Node *current = head, *temp = NULL;

    if (head == NULL) {
        printf("List is empty!\n");
        return;
    }

    // Swap prev and next pointers of each node
    while (current != NULL) {
        temp = current->prev;
        current->prev = current->next;
        current->next = temp;
        current = current->prev; // move to next node (previous in original list)
    }

    // Update head
    if (temp != NULL)
        head = temp->prev;

    printf("List reversed successfully!\n");
}

// Main menu
int main() {
    int choice, data, key;

    while (1) {
        printf("\n--- Doubly Linked List Menu ---\n");
        printf("1. Insert Node at End\n");
        printf("2. Display List\n");
        printf("3. Search Node\n");
        printf("4. Count Nodes\n");
        printf("5. Reverse List\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
        case 1:
            printf("Enter data to insert: ");
            scanf("%d", &data);
            insertEnd(data);
            break;

        case 2:
            displayList();
            break;

        case 3:
            printf("Enter data to search: ");
            scanf("%d", &key);
            searchNode(key);
            break;

        case 4:
            countNodes();
            break;
```

```
            case 5:
                reverseList();
                break;

            case 6:
                printf("Exiting...\n");
                exit(0);

            default:
                printf("Invalid choice!\n");
            }
        }
        return 0;
}
```

Output :

Chaitanya Autade

@chaitanyaautad2

Personal Information

chaitanya.magic@gmail.com

Add your mobile number

India

My Resume

+ Add Resume

Add your resume here

EEO settings

Complete your profile

Add your missing details →

This data will be helpful to auto-fill your job applications

0%

My Badges

My Certifications

Work Experience

+ Add Work Experience