Date:
P. No:

Name :- Sakshi chourasiya.
Subject :- DSA-Lab
Semester :- III rd Sem. (C.S.E) 2nd year.

Index

Sr.no.	Name of the Experiment	Teacher's Sign.	Remarks.
(i)	WAP to find avg in n number of array in C		
(ii)	WAP to Insert a node at the beginning in a Singly linked list.		
(iii)	WAP to create a Single linked list		
(iv)	WAP to Insert the new node at a particular position in singly linked list		
(v)	WAP to delete a node at the beginning in the singly linked list		
(vi)	WAP to delete the node at a particular position in singly linked list.		
(vii)	WAP to delete a node at the end in Singly linked list		
(viii)	WAP to Implement binary search in C language		
(viiii)	WAP to Implementation of queue using linked list.		

P. No. :

(x) WAP to implemention of stack using array.

(xi) WAP to implemention of merge Sort in C.

(xii) WAP to implemention of quick Sort in C.

(xiii) WAP to implemention of Selection Sort in C.

(xiv) WAP to insert a node at the end in singly linked list.

Date:
P. No:

Experiment :- 1

Aim :- WAP to find avg of n number of array in C.

```
#include<stdio.h>
int main()
{
 int a[25], n, i;
 float mean=0, sum=0;
printf("Enter the numbers of terms :");
 Scanf("%d", &n);
printf("\n Enter the numbers :\n");
for(i=1; i<=n; i++)
{
Scanf("%d", &a[i]);
}
for(i=1; i<=n; i++)
{
Sum=Sum+a[i];
avg=Sum/n;
}
 printf("\n mean of entered numbers are : %f", mean);
 return(0);
}
```

Program :-1

Output

Enter the Numbers of terms : 5

Enter the Numbers.

12
65
32
15
24

Mean of entered numbers are : 29.600000

Date:
P. No:

Experiment :- 2
Aim :- WAP to Create a single linked list

```
#include <stdio.h>
#include <stdlib.h>
Struct node
{
int data;
Struct node * next;
};
Struct nod * head, * tail = null;
void addnode (int data)
{
Struct node * new node = (Struct node *) malloc
(size of)
(Struct node));
newnode -> data = data;
newnode -> next = Null;
if (head == Null)
{
head = newNode;
tail = new Node;
}
else
{
tail -> next = new Node;
tail = new node;
}
}
```

```
void display()
{
struct node * current = head;
if(head == NULL)
{ printf("List is empty\n");
return;
}
printf("Nodes of singly linked list:\n");
while (current != NULL)
{
printf("%d", current->data);
current = current -> next;
}
printf("\n");
}
int main()
{
add node(1);
add node(2);
add node(3);
add node(4);

display();

return 0;
}
```

rogram :-2

utput

Nodes of Singly Linked list :

1, 2, 3, 4

Date:
P. No:

Experiment:-3
Aim:- WAP to Insert a node at the begin note in Singly linked list.

```
#include<stdio.h>
#include<stdlib.h>
void begininsert(int);
Struct node
{
int data;
Struct node *next;
};
    int choice, item;
    do
{
  printf("\n Enter the item which you want to
          Insert? \n");
  Scanf("%d", &item);
  begininsert(item);
  printf("\n press 0 to Insert more? \n");
   Scanf("%d", &choice);
}
  while(choice==0);
}
  void begininsert(int item)
  {

Struct node *ptr=(Struct node *) malloc(size of
    Struct node *));
    if(ptr==null)
    {
```

```
printf ("\n OVERFLOW \n");
}
else
{
    ptr -> data = item;
    ptr -> next = head;
    head = ptr;
  printf ("\n Node inserted \n");
    }
}
```

rogram :-3

utput

Enter the item which you want to insert :

12

Node inserted

Press 0 to insert more :

0

Enter the item which you want to insert :

23

Node inserted

Press 0 to insert more :

2

Date:
P. No:

Experiment :- 4
Aim :- WAP to insert a new node at the particular position in singly linked list ?

```
#include <stdio.h>
#include <stdlib.h>
void randominsert(int);
void create (int);
Struct node
{
  int data;
Struct node * next;
};

Struct node * head;
void main()
{
  int choice, item, loc;
  do
{
printf ("\n Enter the item which you wannt to
          insert ? \n");
Scanf ("%d", & item);
If (head == Null)
{
{ int choice, item, loc;
   do
};
  { create (item);
  };
  else
```

```
{
randominsert (item);
}
printf ("\n press 0 to insert more ? \n");
scanf ("%d", &choice);
}
while (choice == 0);
}
void create (int item)
{
struct node * ptr = (struct node *) malloc (size
(struct node *));
if (ptr == NULL)
{
printf ("\n OVERFLOW\n");
}
else
{
ptr -> data = item;
ptr -> next = head;
head = ptr;
printf ("\n Node Inserted \n");
}
}

void orendominsert (int item)
{
Struct node * ptr (struct node *) malloc (sizeof
(struct node));
Struct node * temp;
int i, loc;
```

```
if (ptr == Null)
{
  printf ("In OVERFLOW");
}
else
{
  printf ("Enter the location");
  Scanf ("%d", &loc);
  ptr -> data = item;
  temp = head;
 for (i=0; i<loc; i++)
 {
  temp = temp -> next;
  if (temp == Null)
  {
   printf ("In can't insert \n");
   return;
  }
 }
 ptr -> next = temp -> next;

  temp -> next = ptr;
 printf ("In Node Inserted");

 }

}
```

Program:-4

Output

Enter the item which you want to Insert ?

12

Node Inserted

Press 0 to Insert more!

2

Date :
P. No. :

Experiment :- 5
Aim :- WAP to insert to a node at the end in singly linked list?

```
#include <stdio.h>
#include <stdlib.h>
Struct node
{
int data;
Struct node * next;
} * head;
void createlist (int n);
void insertNodeAtEnd( int data);
void displaylist();
int main()
{
int n, data;
printf (" Enter the toatal number of nodes:");
Scanf (" %d", &n);
Createlist (n);
Printf("\n Data in the list \n");
displaylist();

printf("\n Enter data to insert at end of the List:");
Scanf ("%d", & data);
insert node At End (data);

printf("\n Data in the List \n");
display List();
return 0;
```

```
void createlist (int n)
{
  struct node * newnode, * temp;
  int data, i;

head = (struct node *) malloc (sizeof (struct n
if (head == Null)
{
printf ("Unable to allocate memory.");
}
else
{
printf ("Enter the data of node 1:");
  scanf ("%d", & data);

head → data = data;
head → next = Null;
temp = head;

for (i = 2; i <= n; i++)
{

newnod = (struct node *) malloc (sizeof (struct node));
if (new node == Null)
{
printf ("Unable to allocate memory");
  break;
}
  else {
```

```
{
  printf("Enter the data of node %d:", i);
  scanf("%d", &data);
  newnode->data = data;
  newnode->next = NULL;
  temp->next = newnode;
  temp = temp->next;
}
}
printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
}
}
void insertnodeAtEnd(int data)
{
struct node *newnode, *temp;
newnode = (struct node*) malloc(sizeof(struct node));
if(newnode == NULL)
{
printf("Unable to allocate memory");
}
else
{
newnode->data = data;
newnode->next = NULL;
temp = head;
```

```
while (temp!=Null && temp->next!=Null)
    temp=temp->next;

temp->next=newnode;
printf("DATA INSERTED SUCCESS FULLY \n");
}
}

void displaylist()
{
struct node *temp;
OF(head==Null)
{
printf("List is empty.");
}
else
{
temp=head;
while(temp!=Null)
{
printf("Data=%d\n",temp->data);
  temp=temp->next;
}
}
}
```

output

Enter the to
Enter the
Enter the
Enter the
SINGLY
Data
Data =
Data =
Data
Enter
DATA
Da
D
D

Program: 5

Output

Enter the total number of nodes : 3
Enter the data of node 1 : 10
Enter the data of node 2 : 20
Enter the data of node 3 : 30
SINGLY LINKED LIST SUCCESSFULLY CREATED

Data in the list
Data = 10
Data = 20
Data = 30
Enter data to insert at end of the the list : 40
DATA INSERTED SUCCESSFULLY

Data in the list
Data = 10
Data = 20
Data = 30
Data = 40

Date :
P. No. :

Experiment :- 6

Aim :- WAP to delete a node at a begin node in singly linked list?

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
 int data;
struct node* next;
} *head;
void create List (int n);
void delete first node ();
void displaylist();
int main()
{
 int n, choice;
 printf (" Enter the total number of nodes : ");
 Scanf ("%d", &n);
Crente list (n);
printf ("In Data in the List In");
display List ();
printf ("In press 1 to delete first node : ");
 Scanf ("%d", & choice);
if (choice == 1)
delete first node();
printf ("In Data in the list In");
 display List ();
return 0;
}
void createlist (int n)
{
```

```
struct node * newnode, * temp;
int data, i;
head = (struct node *) malloc (size of (struct node));
if (head == Null)
printf ("Unable to allocate memory.");
}
else
{
printf ("Enter the data of node 1:");
Scanf ("%d", &data);
head -> data = data;
head -> next = Null;
temp = head
for (i = 2; i < n; i++)
{
newnode (struct node *) malloc (sizeof (struct node));
if (newnode == Null)
{
printf ("Unable to allocate memory");
break;
}
else
{
printf ("Enter the data of node %d:", i);
Scanf ("%d", &data);
newnode -> data = data;
newnode -> next = Null;

temp -> next = newnode;
temp = temp -> next;
}
}
printf ("SING
}
void delete fir
{
struct nod
if (head =
{
printf ("
}
else
{
to Dele
head =
printf
free (
printf
}
void
{
st
of C
{
```

```
    }
  }
  printf ("SINGLY LINKED LIST CREATED SUCCESS FULLY)
                                        \n");
  }
}
void deletefirstnode ()
{
  struct node * toDelete;
  if (head == Null)
  {
  printf ("List is already empty.");
  }else
  {
  toDelete = head;
  head = head -> next;
  printf ("\n Data deleted = %d\n", toDelete -> data);
  free (toDelete);
  printf (" SUCCESSfully DELETED FIRST NODE FROM
          LIST\n");
  }
}void displaylist()
{
  struct node * ;
  if(head == Null)
  {
```

```
printf("List is empty.");
}
else
{
 temp = head;
while (temp != NULL)
{
 printf("Data = %d \n", temp -> data);
   temp = temp -> next;
 }
}
}
```

Program 6

output

Enter the total number of nodes : 5

Enter the data of node 1 : 10

Enter the data of node 2 : 20

Enter the data of node 3 : 30

Enter the data of node 4 : 40

Enter the data of node 5 : 50

SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list

Data = 10

Data = 20

Data = 30

Data = 40

Data = 50

press 1 to delete first node : 1

Data deleted = 10

SUCCESSFULLY DELETED FIRST NODE FROM LIST

Data in the List

Data = 20

Data = 30

Data = 40

Data = 50

Date:
P. No:

Experiment :-7
Aim :- WAP to delete a node at the particular position in singly Linked list.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
  int data;
  struct node * next;
};

void push(struct node * head_ref, int new_data)
{
 struct node *new_node =(struct node *)malloc
 (sizeof(struct node));
 new_node->data = new_data;
 new_node -> next = (* head_ref);
 (* head_ref) = new_node
}
void delete node(struct node * *head_ref, int
                                    position)

{
 if (* head_ref == Null)
   return;
   struct node * temp = * head_ref;
   if ( position == 0)

{ * head_ref = temp -> next;
    free (temp);
     return;
  }
```

```
for(int i=0; temp!=Null && i< position-1; i++)
    temp= temp->next;

if (temp == Null || temp->next == Null)
    return;
Struct node * next = temp->next->next;
free(temp->next);
  temp->next = next;
}

void printList (Struct node * node)
{
while (node != Null)
{
printf ("%d", node->data);
  node = node->next;
}
}
int main()
{
push (& head, 7);
push (& head, 1);
push (& head, 9);
push (& head, 2);
push (& head, 8);

puts ("Created Linked List:");
printlist (check);
delete nod (& head, 4);
```

```
puts ("In Li
printList
return
}
```

Date :
P. No. :

puts("In Linked List after Deletion at position 4:");
printList(head);
return 0;
}

Program:7

Output:-

Created Linked List:

8, 2, 3, 1, 7

Linked list after Deletion at position 4:

8, 2, 3, 1

Date:
P. No:

Experiment :- 8
Aim :- WAP to delete a node at the end in Singly linked list.

```
#include <stdio.h>
Struct node
{
  int data;
Struct node * next;
};
Struct node * head; * tail = Null;
void addNode (int data)
{
Struct node * newnode = (Struct node *) malloc (sizeof
(Struct node));
new node -> data = data;
new node -> next = Null;
if (head == Null)
{
  head = newNode;
  tail = newnode;
}
else
{
tail -> next = newNode
tail = newnode;
}
}
  void delete from End()
```

```
{
  if (head == Null)
{
  printf("List is empty\n");
  return;
}
else
{
  if(head != tail)
{
  struct node * current=head;
  while (current -> next != tail)
{
  current = current -> next;
}
  tail = current;
  tail -> next = Null
}
  else
{
  head = tail = Null;
}
}
}
void display()
{
  struct node * current = head;
  if (head == null)
{
```

```
printf("List is empty\n");
return;
}
while(current != NULL)
{
printf("%d ", current->data);
current = current->next;
}
printf("\n");
}
int main()
{
addnode(1);
addnode(2);
addnode(3);
addnode(4);
}
}
}
printf("Original List:\n");
display();
while(head != NULL)
{
deletefromEnd();
printf("Updated List:\n");
display();
}
return 0;
}
```

Program :-8

Output

Original List :

1 , 2 , 3 , 4

Updated List :

1 , 2 , 3

Updated List :

1 , 2

Updated List :

1

Updated List :

List is empty.

Date:
P. No:

Experiment:9

Aim:- WAP to implement binary binary search in C language.

```
#include<stdio.h>
#int iterative Binary Search(int array[], int start
  -index,
int end-index, int element)
{
  while(start-index<=end-index)
{
int middle=start-index+(end-index-start
  if(array[middle]==element)
      return middle;
  if(array[middle]<element)
    start_index=middle+1;
else
    end_index=middle-1;
}
return-1;
}
int main(void)
  {
  intarray[]={1,4,7,9,16,56,70};
int n=7;
int element=16;
intfeude-index=iterativeBinarySearch(array,0,n-1
element);
if(-foudn-index==-1)
{
printf("Element not found in the array");
}
```

```
else
{
printf("Element found at index: %d", found);
}
return 0;
}
```

ogram :-9

output

Element found

Program :-9

output

Element found at Index : 4

Date:
P. No:

Experiment :- 10
Aim :- WAP to implementation of stack using array.

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

int n, top = -1, *stack;
void push(int *)
{
  if (top == n) return;
  Stack [++top] = x;
}
int pop()
{
  if (top == -1) return -1;
  return stack [top--);
}
int peak()
{
if (top == -1) return -1;
  return stack [top];
}
void display()
{
for (int i = top; i > -1; i--)
printf ("%d", stack [i]);
printf ("\n\n");
}
int main()
{
```

P. No.:

```
n=10;
printf ("Initializing the stack with size 10\n\n");
Stark=(int *) malloc (n * sizeof (int));
printf ("Pushing elements into the stack\n\n");
push(1);
push(2);
push(3);

printf("Displaying elements of the Stack -\n");
display();

printf("The top of the stack = %d\n\n", peek());
printf("Pop the top of the Stack = %d\n\n", Pop
printf("Pop the top of the stack = %d\n\n", POP()
printf("Displaying elements of the Stack -\n")

display();

return 0;

}
```

Program:-10

Output

```
Initializing the S
10
Pushing elements
1
2
3
Displaying elem
-
3 2 1
The top of the
Pop the top of
Pop the top
Displaying
-
1
```

Program:-10

Output

Initializing the stack with size

10

Pushing elements into the stack

1
2
3

Displaying elements of the stack

3 2 1

The top of the stack = 3

POP the top of the stack = 3

POP the top of the stack = 2

Displaying elements of the stack

1

Date:
P. No:

Experiment :- 11

Aim :- WAP to Implementation of queue using Linked List.

```
#include <stdio.h>
#include <stdlib.h>
struct QNode
{
int key;
struct QNode *next;
};

struct Queue
{
struct QNode *front, *rear;
};

struct QNode * newNode (int k)
{
struct QNode * temp = (struct QNode *) malloc
(size of (struct QNode));
temp -> key = k;
temp -> next = NULL;
return temp;
}
Struct Queue * createQueue ()
{
Struct Queue * q = (struct Queue *) malloc (size
of (struct Queue));
q -> front = q -> rear = NULL;
return q;
}
```

```
void enque(struct Queue *q, int k);
if(q->rear == NULL)
q->front = q->rear = temp;
return;
}
q->rear->next = temp;
q->rear = temp;
}
voidde Queue(struct queue *q)
{
if(q->front == NULL)
return;
Struct Qnode *temp = q->front;
q->front = q->front->next

if(q->front == NULL)
q->rear = NULL;
free(temp);
}

int main()
{
Struct Queue *q = Create Queue();
enQueue(q,10);
enQueue(q,20);
deQueue(q);
deQueue(q);
enQueue(q,30);
enQueue(q,40);
enQueue(q,50);
deQueue(q);
```

printf(
printf
return

```
printf("Queue fornt : %d \n", q -> fornt -> key);
printf("Queue Rear : %d ", q -> rear -> key);
return 0;
```

ogram :- 11

output

Queue foront : 40

Queue Rear : 50

Date :
P. No. :

Experiment :- 12
Aim :- WAP to implementation of merge sort in c.

```
#include <stdio.h>
#define max 10

int a[11] = {10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0};
int b[10];

void merging(int low, int mid, int high)
{
int l1, l2, i;
for(l1 = low, l2 = mid + 1, i = low; l1 <= mid &&
l2 <= high; i++)

if(a[l1] <= a[l2])
  b[i] = a[l1++];
else
   b[i] = a[l2++];
}

while(l1 <= mid)
    b[i++] = a[l1++];

while(l2 <= high)
    b(i++) = a[l2++];

for(i = low; i <= high; i++)
  a[i] = b[i]
}
```

```
void Sort (int low, int high)
{
 int mid;
if (low<high)
{
mid = (low+high)/2;
Sort(low, mid);
Sort (mid+1, high);
mergeng (low, mid, high);
}
else
{
 return;
}
}
int main()
{
 int i;
printf ("List before Sorting \n");
for(i=0; i<=max; i++)
 printf("%d", a[i];
Sort (0, max);
printf ("\n List after sorting \n");
for(i=0; i<=max; i++)
printf ("%d", a[i]);
}
```

Program-12

output

List by core sorting

10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0

List after sorting

0, 10, 14, 19, 26, 27, 31, 35, 42, 44

Date :
P. No. :

Experiment :- 13
Aim :- WAP to implementation of quick Sort in C.

```
#include <stdio.h>
void quicksort(int number[25], int first, int last)
{
  int i, j, pivot, temp;
  if(   first < last)
  {
    pivot = first;
      i = first;
      j = last;

  while (i<j)
  {
  while(number[i]<=number[pivot]&&i<last)
   i++;
  while(number[j]>number[pivot])
   j--;
   if(i<j)
   {

    temp=number[i];
    number[i] = number[j];
    number[j]=temp;
  }

   }
   temp = number[pivot];
  number[pivot] =number[j];
  number[j] =temp;
```

```
quicksort(number, first, j-1);
quicksort(number, j+1, last);
}
}
int main()
{
int i, count, number[25];
printf("How many elements are you going to enter?: ");
scanf("%d", &count);

printf("Enter %d elements", count);
for(i=0; i<count; i++)
scanf("%d", &number[i]);

quicksort(number, 0, count-1);
printf("Order of Sorted elements: ");
for(i=0; i<count; i++)
printf("%d", number[i]);

return 0;
}
```

utput

How many e

Enter 5 ele

order of

Program :-13

Output

How many elements are u going to enter ? : 5

Enter 5 elements : 88 0 -9 98 27

order of sorted elements :- -9 0 27 88 98.

Date :
P. No. :

Experiment :- 14
Aim :- WAP to implementation of Selection Sort in C;

```
#include <stdio.h>
int main ()
{
int a[100], n, i, j, position, swap;
printf ("Enter number of elements n");
scanf ("%d", &n);
printf ("Enter %d numbers n", n);
for (i=0; i<n; i++)
scanf ("%d", &a[i]);
for (i=0; i<n-1; i++)
{
position = i;
for (j = i+j; j<n; j++)
{
if (a[position] > a[j])
position = j;
}
if (position != i)
{
swap = a[i];
a[i] = a[position];
a[position] = swap;
}
}
printf ("sorted Array : n");
for (i=0; i<n; i++)
```

Date :
P. No. :

```
printf ("%d\n", q[i]);
return 0;
}
```

Program -
output
Enter number
4
Enter 4 num
4 2 7 1
Sorted a
1 2 4 7
Gira

ogram - 14

utput

Enter number of elements

4

Enter 4 numbers

4
2
7
1

Sorted array

1
2
4
7

Program exited with code :-