PRACTICAL – 1

```
#include<stdio.h>
void swap(int,int);
int main()
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
int a,b;
printf("\nEnter a Value of A=");
scanf("%d",&a);
printf("\nEnter a Value of B=");
scanf("%d",&b);
swap_by_value(a,b);
swap_by_refrence(&a,&b);
}
void swap_by_value(int p,int q)
{
int tmp;
tmp=p;
p=q;
q=tmp;
printf("\nCall By Value\n");
printf("\nNew Values After Swap:");
printf("A=%d B=%d",p,q);
}
void swap_by_refrence(int *p , int *q)
{
int tmp;
tmp=*p;

*p=*q;
*q=tmp;
printf("\n\nCall By Refrence\n");
printf("\nNew Values After Swap:");
printf("A=%d B=%d",*p,*q);
}
```

```
Name: Yash Tripathi
En.no.: 210410107140

Enter a Value of A=5
Enter a Value of B=7
Call By Value

New Values After Swap:A=7 B=5

Call By Refrence

New Values After Swap:A=7 B=5
```

PRACTICAL – 2

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");

int *ptr,*ptr1;
int n,i,sum=0;
printf("Enter Number of Elements: ");
scanf("%d",&n);
ptr=(int*)malloc(n*sizeof(int));
ptr1=(int*)malloc(n*sizeof(int));
printf("Enter Elements of Array: \n");
for(i=0;i<n;++i)
{
scanf("%d",ptr+i);
sum+=*(ptr+i);
}
printf("Sum=%d\n",sum);
printf("\nEven Number in Array\n");
for(i=0;i<n;++i)
{
if(*(ptr+i)%2==0)
{
printf("%d\n",*(ptr+i));

}
}
free(ptr);
return 0;
}
```

```
Name: Yash Tripathi
En.no.: 210410107140
Enter Number of Elements: 4
Enter Elements of Array:
5
3
7
2
Sum=17

Even Number in Array
2
```

PRACTICAL – 3

```
#include<stdio.h>
#define MAX 50
int a[MAX], top = -1;
void push();
void pop();
void peep();
void change();
void display();
void main()
{
int ch;
while(1)
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
printf("\n1. PUSH or INSERT");
printf("\n2. POP or DELETE");
printf("\n3. PEEP or SEARCH");
printf("\n4. CHANGE or UPDATE");
printf("\n5. Display");
printf("\n6. End program");
printf("\nEnter Choice : ");
scanf("%d",&ch);
switch(ch)
{
case 1:
{
push();
break;

}
case 2:
{
pop();
break;
}
case 3:
{
peep();
```

```
Name: Yash Tripathi
En.no.: 210410107140

1. PUSH or INSERT
2. POP or DELETE
3. PEEP or SEARCH
4. CHANGE or UPDATE
5. Display
6. End program
Enter Choice : 1
Enter the element : 3
Name: Yash Tripathi
En.no.: 210410107140

1. PUSH or INSERT
2. POP or DELETE
3. PEEP or SEARCH
4. CHANGE or UPDATE
5. Display
6. End program
```

```
break;
}
case 4:
{
change();
break;
}
case 5:
{
display();
break;
}
case 6:
{
exit(0);
}
default:
{

printf("\ninvalid choice !!!");

}
}
}
getch();
}
void push(){
int data;
if(top==MAX-1)
{
printf("\noverflow or stack is full !!!");

}
else
{
printf("\nEnter the element : ");
scanf("%d",&data);
top++;
a[top]=data;
}
}
```

```
Enter Choice : 5
Elemets :
3
Name: Yash Tripathi
En.no.: 210410107140

1. PUSH or INSERT
2. POP or DELETE
3. PEEP or SEARCH
4. CHANGE or UPDATE
5. Display
6. End program
Enter Choice : 2
Deleted element : 3
Name: Yash Tripathi
En.no.: 210410107140

1. PUSH or INSERT
2. POP or DELETE
3. PEEP or SEARCH
4. CHANGE or UPDATE
```

```
void pop()
{
if(top==-1)
{
printf("\nStack is underflow");
}
else
{
printf("\nDeleted element : %d",a[top]);
top--;
}
}
void display()
{
int i;
if(top>=0)
{
printf("\nElemets : ");
for(i=top; i>=0; i--)
{

printf("\n%d",a[i]);

}
}
else
{
printf("\nThe Stack is Empty");
}
}

void peep()
{
int p;
printf("\nEnter the position : ");
scanf("%d",&p);
if(top-p<=-1)
{
printf("\nInvalid Position");
}
else
```

```
{
printf("\nThe Elements is : %d",a[top-p]);
}
}
void change()
{
int v1,v2;
printf("\nEnter Position for change : ");
scanf("%d",&v1);
printf("\nEneter the Number for change : ");
scanf("%d",&v2);
if(top-v1<=-1)
{
printf("\nSTACK is overflow !!!");
}
else
{
a[top-v1]=v2;
printf("\nCHANGE successfull !!!");
}
}
```

```
Name: Yash Tripathi
En.no.: 210410107140

1. PUSH or INSERT
2. POP or DELETE
3. PEEP or SEARCH
4. CHANGE or UPDATE
5. Display
6. End program
Enter Choice : 6
```

PRACTICAL – 4

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>
#define SIZE 100
char stack[SIZE];
int top = -1;
void push(char item)
{
if(top >= SIZE-1)
{
printf("\nStack Overflow.");
}
else
{
top = top+1;
stack[top] = item;
}
}
char pop()
{
char item ;
if(top <0)
{

printf("stack under flow: invalid infix expression");
getchar();
/* underflow may occur for invalid expression */
/* where ( and ) are not matched */
exit(1);
}
else
{
item = stack[top];
top = top-1;
return(item);
}
}
int is_operator(char symbol)
```

```
{
if(symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol
=='-')
{
return 1;
}
else
{
return 0;
}
}
int precedence(char symbol)
{
if(symbol == '^')
{
return(3);
}
else if(symbol == '*' || symbol == '/')
{

return(2);
}
else if(symbol == '+' || symbol == '-')
{
return(1);
}
else
{
return(0);
}
}
void InfixToPostfix(char infix_exp[], char postfix_exp[])
{
int i, j;
char item;
char x;
push('(');
strcat(infix_exp,")");
i=0;
j=0;
item=infix_exp[i];
```

```
while(item != '\0')
{
if(item == '(')
{
push(item);
}
else if( isdigit(item) || isalpha(item))
{
postfix_exp[j] = item;
j++;

}
else if(is_operator(item) == 1)
{
x=pop();
while(is_operator(x) == 1 && precedence(x)>= precedence(item))
{
postfix_exp[j] = x;
j++;
x = pop();
}
push(x);
push(item);
}
else if(item == ')')
{
x = pop();
while(x != '(')
{
postfix_exp[j] = x;
j++;
x = pop();
}
}
else
{
printf("\nInvalid infix Expression.\n");
getchar();
exit(1);
}
i++;
```

```
item = infix_exp[i];
}

if(top>0)
{
printf("\nInvalid infix Expression.\n");
getchar();
exit(1);
}
if(top>0)
{
printf("\nInvalid infix Expression.\n");
getchar();
exit(1);
}
postfix_exp[j] = '\0';
}
int main()
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
char infix[SIZE], postfix[SIZE];
printf("\nEnter Infix expression : ");
gets(infix);
InfixToPostfix(infix,postfix);
printf("Postfix Expression: ");
puts(postfix);
return 0;
}
```

```
Name: Yash Tripathi
En.no.: 210410107140

Enter Infix expression : A+B*C/D+E-F
Postfix Expression: ABC*D/+E+F-
```

PRACTICAL – 5

```
#include<stdio.h>
#include<stdlib.h>
#define n 5
int main()
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
int queue[n],ch=1,front=0,rear=0,i,j=1,x=n,temp;
printf("\n--------------------\n");
printf("Queue using Array");
printf("\n--------------------\n");
printf("\n1.Insertion \n2.Deletion \n3.Display \n4.Exit\n");
printf("\n--------------------\n");
while(ch)
{
printf("\nEnter the Choice:");
scanf("%d",&ch);
switch(ch)
{
case 1:
if(rear==x)
printf("\n Queue is Full");
else
{
printf("\n Enter no %d:",j++);
scanf("%d",&temp);
queue[rear++]=temp;
}
break;

case 2:
if(front==rear)
{
printf("\n Queue is empty");
}
else
{
printf("\n Deleted Element is %d",queue[front++]);
x++;
}
```

```
Name: Yash Tripathi
En.no.: 210410107140

--------------------
Queue using Array
--------------------

1.Insertion
2.Deletion
3.Display
4.Exit

--------------------

Enter the Choice:1
Enter no 1:2
Enter the Choice:1
Enter no 2:5
Enter the Choice:3
Queue Elements are:
```

```
break;
case 3:
printf("\n Queue Elements are:\n ");
if(front==rear)
printf("\n Queue is Empty");
else
{
for(i=front; i<rear; i++)
{
printf("%d",queue[i]);
printf("\n");
}
break;
case 4:
exit(0);
default:
printf("Wrong Choice: please see the options");
}
}
}
}
```

```
Queue Elements are:
 2
5

Enter the Choice:2
Deleted Element is 2
Enter the Choice:3
Queue Elements are:
 5

Enter the Choice:4
```

PRACTICAL – 6

```
#include<stdlib.h>
#include<stdio.h>
#define max 5
int front=-1,rear=-1;
int CQueue[max];
void insert();
int delete1();
void display();
int main()
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
int ch,no;
printf("\n1. Insert");
printf("\n2. Delete");
printf("\n3. Display");
printf("\n4. EXIT");
while(ch)
{
printf("\nEnter the Choice:");
scanf("%d",&ch);
switch(ch)
{
case 1:
insert();
break;

case 2:
no=delete1();
break;
case 3:
display();
break;
case 4:
exit(1);
default:
printf("\nInvalid Choice !!\n");
}
}
}
```

```
Name: Yash Tripathi
En.no.: 210410107140

1. Insert
2. Delete
3. Display
4. EXIT
Enter the Choice:1
Enter a number to Insert :4
Enter the Choice:1
Enter a number to Insert :3
Enter the Choice:1
Enter a number to Insert :9
Enter the Choice:1
Enter a number to Insert :8
Enter the Choice:1
Enter a number to Insert :3
Enter the Choice:1
Circular Queue Is Full !
```

```
void insert()
{
int no;
if((front ==0 && rear == max-1) || front == rear+1)
{
printf("\nCircular Queue Is Full !\n");
return;
}
printf("\nEnter a number to Insert :");
scanf("%d",&no);
if(front==-1)
front=front+1;
if(rear==max-1)
rear=0;
else

rear=rear+1;
CQueue[rear]=no;
}
int delete1()
{

int e;
if(front==-1)
{
printf("\nThe Circular Queue is Empty !!\n");
}
e=CQueue[front];
if(front==max-1)
front=0;
else if(front==rear)
{
front=-1;
rear=-1;
}
else front=front+1;
printf("\n%d was deleted !\n",e);
return e;
}
void display()
{
```

```
int i;
if(front==-1)
{
printf("\nThe Circular Queue is Empty ! Nothing To Display !!\n");
return;
}
i=front;
printf("The elements are");
if(front<=rear)
{
printf("\n\n");
while(i<=rear)
printf("%d ",CQueue[i++]);
printf("\n");

}
else
{
printf("\n\n");
while(i<=max-1)
printf("%d ",CQueue[i++]) ;
i=0;
while(i<=rear)
printf("%d ",CQueue[i++]);
printf("\n");
}
return 0;
}
```

```
Enter the Choice:3
The elements are

4 3 9 8 3

Enter the Choice:2
4 was deleted !

Enter the Choice:4
```

PRACTICAL – 7

Insert a node at the front of the linked list

```
#include <stdio.h>
#include <stdlib.h>
struct node {
int data;
struct node *next;
} *head;
void initialize(){
head = NULL;
}

void insertAtFront(int num) {
struct node* newNode = (struct node*) malloc(sizeof(struct node));
newNode->data = num;

newNode->next = head;
head = newNode;
printf("Inserted Element : %d\n", num);
}

void printLinkedList(struct node *nodePtr) {
printf("\nLinked List\n");
while (nodePtr != NULL) {
printf("%d", nodePtr->data);
nodePtr = nodePtr->next;
if(nodePtr != NULL)
printf("-->");
}
}
int main() {
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
initialize();
insertAtFront(2);
insertAtFront(4);
insertAtFront(5);
insertAtFront(9);
printLinkedList(head);

return 0;}
```

```
Name: Yash Tripathi
En.no.: 210410107140
Inserted Element : 2
Inserted Element : 4
Inserted Element : 5
Inserted Element : 9

Linked List
9-->5-->4-->2
```

Insert a node at the end of the linked list

```
Name: Yash Tripathi
En.no.: 210410107140
After Insertion At End
Linked List
2-->10
```

```
#include <stdio.h>
#include <stdlib.h>
struct node {
int data;
struct node *next;
} *head;
void initialize(){
head = NULL;
}
void insertAtFront(int num) {
struct node* newNode = (struct node*) malloc(sizeof(struct node));
newNode->data = num;
newNode->next = head;
head = newNode;
}
void insertAtEnd(struct node* head, int num){
if (head == NULL) {
printf("Error : Invalid node pointer !!!\n");
return;
}
struct node* newNode =(struct node*) malloc(sizeof(struct node));
newNode->data = num;
newNode->next = NULL;
while(head->next != NULL)
head = head->next;
head->next = newNode;
}
void printLinkedList(struct node *nodePtr) {
printf("\nLinked List\n");
while (nodePtr != NULL) {
printf("%d", nodePtr->data);
nodePtr = nodePtr->next;
if(nodePtr != NULL)
printf("-->");
}
}
int main() {
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
initialize();
```

```
insertAtFront(2);
insertAtEnd(head, 10);
printf("\n\nAfter Insertion At End\n");
printLinkedList(head);
return 0;}
```

Insert a node such that linked list is in ascending order.(according to info. Field)

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
int data;
struct Node* next;
};
void sortedInsert(struct Node** head_ref,struct Node* new_node)
{
struct Node* current;
if (*head_ref == NULL || (*head_ref)->data>= new_node->data)
{
new_node->next = *head_ref;
*head_ref = new_node;
}
else {
current = *head_ref;
while (current->next != NULL && current->next->data <new_node->data)
{current = current->next;}
new_node->next = current->next;
current->next = new_node;}}
struct Node* newNode(int new_data)
{
struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
new_node->data = new_data;
new_node->next = NULL;
return new_node;
}
void printList(struct Node* head)
{
struct Node* temp = head;
while (temp != NULL) {
printf("%d ", temp->data);
temp = temp->next;}}
int main()
```

```
Name: Yash Tripathi
En.no.: 210410107140
Created Linked List
1 3 5 7 9 10
```

```
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
struct Node* head = NULL;
struct Node* new_node = newNode(5);
sortedInsert(&head, new_node);
new_node = newNode(10);
sortedInsert(&head, new_node);
new_node = newNode(7);
sortedInsert(&head, new_node);
new_node = newNode(3);
sortedInsert(&head, new_node);
new_node = newNode(1);
sortedInsert(&head, new_node);
new_node = newNode(9);
sortedInsert(&head, new_node);
printf("\n Created Linked List\n");
printList(head);
return 0;}
```

Delete a first node of the linked list.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int num;
struct node *nextptr;
}*stnode;
void createNodeList(int n);
void FirstNodeDeletion();
void displayList();
int main()
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
int n,num,pos;
printf("\nLinked List : Delete first node of Singly Linked List\n");
printf("----------------------------------------------------------\n");
printf(" Input the number of nodes : ");
scanf("%d", &n);
createNodeList(n);
printf("\n Data entered in the list are : \n");
```

```
displayList();
FirstNodeDeletion();
printf("\n Data, after deletion of first node : \n");
displayList();
return 0;
}
void createNodeList(int n)
{
struct node *fnNode, *tmp;

int num, i;
stnode = (struct node *)malloc(sizeof(struct node));
if(stnode == NULL)
{
printf(" Memory can not be allocated.");
}
else
{
printf(" Input data for node 1 : ");
scanf("%d", &num);
stnode-> num = num;
stnode-> nextptr = NULL;
tmp = stnode;
for(i=2; i<=n; i++)
{
fnNode = (struct node *)malloc(sizeof(struct node));
if(fnNode == NULL)
{
printf(" Memory can not be allocated.");
break;
}
else
{
printf(" Input data for node %d : ", i);
scanf(" %d", &num);
fnNode->num = num;
fnNode->nextptr = NULL;
tmp->nextptr = fnNode;
tmp = tmp->nextptr;}}}}
void FirstNodeDeletion()
{
```

```
struct node *toDelptr;
if(stnode == NULL)
{
printf(" There are no node in the list.");
}
else
{
toDelptr = stnode;
stnode = stnode->nextptr;
printf("\n Data of node 1being deleted is:%d\n",toDelptr->num);
free(toDelptr);}}
void displayList()
{
struct node *tmp;
if(stnode == NULL)
{
printf(" No data found in the list.");
}
else
{
tmp = stnode;
while(tmp != NULL)
{
printf(" Data = %d\n", tmp->num);
tmp=tmp->nextptr;
}}}
```

```
Name: Yash Tripathi
En.no.: 210410107140

Linked List : Delete first node of Singly Linked List
-------------------|----------------------------------------
 Input the number of nodes : 3
 Input data for node 1 : 2
 Input data for node 2 : 8
 Input data for node 3 : 3
 Data entered in the list are :
 Data = 2
 Data = 8
 Data = 3

 Data of node 1being deleted is:2

 Data, after deletion of first node :
 Data = 8
 Data = 3
```

Delete a node before specified position.

```
#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 deleteNode(struct Node **head_ref, int position)
{
if (*head_ref == NULL)
return;
struct Node* temp = *head_ref;
position--;
if (position == 0)
{
*head_ref = temp->next;
free(temp);
return;
}
int i;
for (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()
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
struct Node* head = NULL;
push(&head, 7);
push(&head, 1);
push(&head, 3);
push(&head, 2);
push(&head, 8);
puts("Created Linked List: ");
printList(head);
deleteNode(&head, 2);
puts("\nLinked List after Deletion");
printList(head);
return 0;}
```

```
Name: Yash Tripathi
En.no.: 210410107140
Created Linked List:
 8  2  3  1  7
Linked List after Deletion
 8  3  1  7
```

Delete a node after specified position.

```
#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 deleteNode(struct Node **head_ref, int position)
{
if (*head_ref == NULL)
return;
struct Node* temp = *head_ref;
position++;
if (position == 0)
```

```
{
*head_ref = temp->next;
free(temp);
return;
}
int i;
for (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()
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
struct Node* head = NULL;
push(&head, 7);
push(&head, 1);
push(&head, 3);
push(&head, 2);
push(&head, 8);
puts("Created Linked List: ");
printList(head);
deleteNode(&head, 2);
puts("\nLinked List after Deletion");
printList(head);
return 0;}
```

```
Name: Yash Tripathi
En.no.: 210410107140
Created Linked List:
 8  2  3  1  7
Linked List after Deletion
 8  2  3  7
```

PRACTICAL – 8

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
int data;
struct Node *next;
}*top=NULL;
void push(int x)
{
struct Node *t;
t=(struct Node*)malloc(sizeof(struct Node));
if(t==NULL)
printf("stack is full\n");
else
{
t->data=x;
t->next=top;
top=t;
printf("%d is inserted\n",t->data);
}
}
int pop()
{
struct Node *t;
int x=-1;
if(top==NULL)
printf("Stack is Empty\n");
else
{
t=top;
top=top->next;
x=t->data;
printf("%d is Deleted\n",t->data);
free(t);
}
return x;
}
void Display()
{
printf("Elemnts\n");
```

```
struct Node *p;
p=top;
while(p!=NULL)
{
printf("%d ",p->data);
p=p->next;
}
printf("\n");
}
int main()
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
push(10);
push(20);
push(30);
Display();
pop();
return 0;}
```

```
Name: Yash Tripathi
En.no.: 210410107140
10 is inserted
20 is inserted
30 is inserted
Elemnts
30 20 10
30 is Deleted
```

PRACTICAL – 9

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
int data;
struct Node *next;
}*front=NULL,*rear=NULL;
void enqueue(int x)
{
struct Node *t;
t=(struct Node*)malloc(sizeof(struct Node));
if(t==NULL)
printf("Queue is FUll\n");
else
{
t->data=x;
printf("%d is inserted\n",t->data);
t->next=NULL;
if(front==NULL)
front=rear=t;
else
{
rear->next=t;
rear=t;}}}
int dequeue()
{
int x=-1;
struct Node* t;
if(front==NULL)
printf("Queue is Empty\n");
else
{
x=front->data;
t=front;
front=front->next;
printf("%d is Deleted\n",t->data);
free(t);
}
return x;
}
```

```
void Display()
{
struct Node *p=front;
printf("Elements\n");
while(p)
{
printf("%d ",p->data);
p=p->next;
}
printf("\n");
}
int main()
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
enqueue(10);
enqueue(20);
enqueue(30);
enqueue(40);
enqueue(50);
Display();
dequeue();
Display();
return 0;}
```

```
Name: Yash Tripathi
En.no.: 210410107140
10 is inserted
20 is inserted
30 is inserted
40 is inserted
50 is inserted
Elements
10 20 30 40 50
10 is Deleted
Elements
20 30 40 50
```

PRACTICAL – 10

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
};
struct node *temp,*newnode,*tail;
int count;
struct node *head=NULL;
void create()
{
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter data:- ");
scanf("%d",&newnode->data);
newnode->next=0;
if(head==0)
{
head=tail=newnode;
}
else
{
tail->next=newnode;
tail=newnode;
}

tail->next=head;
count++;
}
void display()
{
if(tail==0)
printf("linkedlist is empty");
else
{
temp=tail->next;
while(temp->next!=tail->next)
{
printf("%d\t",temp->data);
```

```
Name: Yash Tripathi
En.no.: 210410107140
following operations you can perform on the linked list:-
1. create
2. display
3. insert_at_begin
4. insert_at_end
5.insert_after_position
6. insert_before_position
7.delete_first_node
8.delete_last_node
9. delete_specific_node
10. getlength
11.search_node
12. exit
-------------------------------------------------
enter your choice:- 1
enter data:- 10
continue(Y/N):- y
```

```
temp=temp->next;
}
printf("%d\n",temp->data);
}
}
void insert_at_begin()
{
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter data:- ");
scanf("%d",&newnode->data);
newnode->next=0;
if(tail==0)
{
tail=newnode;
tail->next=newnode;
}
else
{
newnode->next=tail->next;
tail->next=newnode;
}
count++;

display();
}
void insert_at_end()
{
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter data:- ");
scanf("%d",&newnode->data);
newnode->next=0;
if(tail==0)
{
tail=newnode;
tail->next=newnode;
}
else
{
newnode->next=tail->next;
tail->next=newnode;
tail=newnode;
```

```
}
count++;
display();
}
void insert_before_position()
{
int pos,i=1;
printf("before which position you want to insert:-\n");
scanf("%d",&pos);
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter data:- ");
scanf("%d",&newnode->data);
newnode->next=0;
if(pos<1 || pos>count+1)
printf("invalid position");
else
{

temp=tail->next;
while(i<pos-1)
{
temp=temp->next;
i++;
}
newnode->next=temp->next;
temp->next=newnode;
}
count++;
display();
}
void insert_after_position()
{
int pos,i=1;
printf("after which position you want to insert:-\n");
scanf("%d",&pos);
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter data:- ");
scanf("%d",&newnode->data);
newnode->next=0;
if(pos<1 || pos>count+1)
printf("invalid position");
```

```
1. create
2. display
3. insert_at_begin
4. insert_at_end
5. insert_after_position
6. insert_before_position
7.delete_first_node
8. delete_last_node
9. delete_specific_node
10. getlength
11. search_node
12. exit
-------------------------------------------------
enter your choice:- 2
10

continue(Y/N):- y
```

```
else
{
temp=tail->next;
while(i<pos)
{
temp=temp->next;
i++;
}
newnode->next=temp->next;
temp->next=newnode;
}
count++;

display();
}
void delete_first_node()
{
temp=tail->next;
if(tail==0)
printf("linked list is empty");
else
{
tail->next=temp->next;
free(temp);
}
count--;
display();
}
void delete_last_node()
{
struct node *prenode;
temp=tail->next;
if(tail==0)
printf("linked list is empty");
else
{
while(temp!=tail)
{
prenode=temp;
temp=temp->next;
}
```

```
following operations you can perform on the linked list:-

1. create
2. display
3. insert_at_begin
4. insert_at_end
5. insert_after_position
6. insert_before_position
7.delete_first_node
8. delete_last_node
9. delete_specific_node
10. getlength
11. search_node
12. exit
-------------------------------------------------
enter your choice:- 3
enter data:- 50
50      10

 continue(Y/N):- y
```

```
prenode->next=temp->next;
tail=prenode;
free(temp);
}
count--;
display();
}

void delete_specific_node()
{
int pos,i=1;
struct node *prenode;
temp=tail->next;
printf("enter the position which you want to delete:-\n");
scanf("%d",&pos);
if(pos<1 || pos>count)
printf("invalid position");
else if(tail==0)
printf("linked list is empty");
else
{
while(i<pos)
{
prenode=temp;
temp=temp->next;
i++;
}
prenode->next=temp->next;
free(temp);
}
count--;
display();
}
void getlength()
{
if(tail==0)
printf("linkedlist is empty");
else
{
printf("length is %d",count);
}
```

```
following operations you can perform on the linked list:-

1. create
2. display
3. insert_at_begin
4. insert_at_end
5. insert_after_position
6. insert_before_position
7.delete_first_node
8. delete_last_node
9. delete_specific_node
10. getlength
11. search_node
12. exit
------------------------------------------------
enter your choice:- 10
length is 2
 continue(Y/N):- y
```

```
}
void search_node()

{
struct node *temp;
int pos,i;
temp=tail->next;
printf("Enter position which you want to search:- ");
scanf("%d",&pos);
if(tail==0)
printf("linkedlist is empty");
else if(pos<1 || pos>count)
printf("invalid position");
else
{
for(i=1;i<pos;i++)
temp=temp->next;
printf("data in the node which you are finding is %d",temp->data);
}
}
void main()
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
int choice;
char ch;
do
{
printf("\nfollowing operations you can perform on the linked list:- \n");
printf("\n1. create\n2. display\n3. insert_at_begin\n4. insert_at_end\
n5.insert_after_position\n6. insert_before_position\n7.delete_first_node\
n8.delete_last_node\n9. delete_specific_node\n10. getlength\n11.search_node\
n12. exit");
printf("\n-----------------------------------------------");
printf("\nenter your choice:- ");
scanf("%d",&choice);
switch(choice)
{
case 1:create();
break;
```

```
case 2:display();
break;
case 3:insert_at_begin();
break;
case 4:insert_at_end();
break;
case 5:insert_after_position();
break;
case 6:insert_before_position();
break;
case 7:delete_first_node();
break;
case 8:delete_last_node();
break;
case 9:delete_specific_node();
break;
case 10:getlength();
break;
case 11:search_node();
break;
case 12:exit(0);
break;
default:printf("enter valid input");
break;
}
printf("\n continue(Y/N):- ");
fflush(stdin);
scanf("%c",&ch);
}while(ch=='Y' || ch=='y');
getch();
}
```

```
1. create
2. display
3. insert_at_begin
4. insert_at_end
5. insert_after_position
6. insert_before_position
7.delete_first_node
8. delete_last_node
9. delete_specific_node
10. getlength
11. search_node
12. exit
-------------------------------------------------
enter your choice:- 11
Enter position which you want to search:- 2
data in the node which you are finding is 10
 continue(Y/N):-
```

PRACTICAL – 11

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
struct node *prev;
};
struct node *tail,*head=0,*newnode;
int count=0;
void create()
{
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter data:-\ ");
scanf("%d",&newnode->data);
newnode->prev=0;
newnode->next=0;
if(head==0)
{
head=tail=newnode;
}
else
{
newnode->prev=tail;
tail->next=newnode;

tail=newnode;
}
count++;
}
void display()
{
struct node *temp;
temp=head;
printf("data in the list are as follow\n");
if(head==0)
printf("linkedlist is empty");
else
{
```

```
Name: Yash Tripathi
En.no.: 210410107140

following operations you can perform on the linked list:-

1. create
2. display
3. insert_at_begin
4. insert_at_end
5.insert_after_position
6. insert_before_position
7. delete_first_node
8.delete_last_node
9. delete_specific_node
10. getlength
11.search_node
12. exit
------------------------------------------------
enter your choice:- 1
enter data:- 10
continue(Y/N):- y
```

```
while(temp!=0)
{
printf("%d\t",temp->data);
temp=temp->next;
}
}
}
void insert_at_begin()
{
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter data:- ");
scanf("%d",&newnode->data);
if(head==NULL)
printf("linkedlist is empty");
else
{
head->prev=newnode;
newnode->next=head;
head=newnode;
}
count++;
display();

}
void insert_at_end()
{
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter data:- ");
scanf("%d",&newnode->data);
if(head==NULL)
printf("linkedlist is empty");
else
{
tail->next=newnode;
newnode->prev=tail;
tail=newnode;
}
count++;
display();
}
void insert_after_position()
```

```
{
int pos,i=1;
struct node *temp;
temp=head;
newnode=(struct node*)malloc(sizeof(struct node));
printf("after which position you want to insert:- ");
scanf("%d",&pos);
printf("enter data:- ");
scanf("%d",&newnode->data);
if(head==NULL)
printf("linkedlist is empty");
else if(pos<1 || pos>count)
printf("invalid position");
else
{
while(i<pos)
{

temp=temp->next;
i++;
}
newnode->prev=temp;
newnode->next=temp->next;
temp->next=newnode;
newnode->next->prev=newnode;
}
count++;
display();
}
void insert_before_position()
{
int pos,i=1;
struct node *temp;
temp=head;
newnode=(struct node*)malloc(sizeof(struct node));
printf("before which position you want to insert:- ");
scanf("%d",&pos);
printf("enter data:- ");
scanf("%d",&newnode->data);
if(head==NULL)
printf("linkedlist is empty");
```

```
else if(pos<1 || pos>count)
printf("invalid position");
else
{
while(i<pos-1)
{
temp=temp->next;
i++;
}
newnode->prev=temp;
newnode->next=temp->next;
temp->next=newnode;

newnode->next->prev=newnode;
}
count++;
display();
}
void getlength()
{
if(head==NULL)
printf("linkedlist is empty");
printf("length is %d",count);
}
void delete_first_node()
{
struct node *temp;
temp=head;
if(head==NULL)
printf("linkedlist is empty");
else
{
head=head->next;
head->prev=0;
free(temp);
}
count--;
display();
}
void delete_last_node()
{
```

```
struct node *temp;
temp=head;
if(head==NULL)
printf("linkedlist is empty");
else
{
while(temp!=tail)

temp=temp->next;
temp->prev->next=0;
tail=temp->prev;
free(temp);
}
count--;
display();
}
void delete_specific_node()
{
int i=1,pos;
struct node *temp;
temp=head;
printf("enter position which you want to delete:-\n");
scanf("%d",&pos);
if(head==NULL)
printf("linkedlist is empty");
else if(pos<1 || pos>count)
printf("invalid position");
else
{
while(i<pos)
{
temp=temp->next;
i++;
}
temp->prev->next=temp->next;
temp->next->prev=temp->prev;
free(temp);
}
count--;
display();
}
```

```
void search_node()
{

struct node *temp;
int pos,i;
temp=head;
printf("Enter position which you want to search:- ");
scanf("%d",&pos);
if(head==NULL)
printf("linkedlist is empty");
else if(pos<1 || pos>count)
printf("invalid position");
else
{
for(i=1;i<pos;i++)
temp=temp->next;
printf("data in the node which you are finding is %d",temp->data);
}
}
void main()
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
int choice;
char ch;
printf("\nfollowing operations you can perform on the linked list:- \n");
printf("\n1. create\n2. display\n3. insert_at_begin\n4. insert_at_end\
n5.insert_after_position\n6. insert_before_position\n7. delete_first_node\
n8.delete_last_node\n9. delete_specific_node\n10. getlength\n11.search_node\
n12. exit");
printf("\n-----------------------------------------------");
do
{
printf("\nenter your choice:- ");
scanf("%d",&choice);
switch(choice)
{
case 1:create();
break;

case 2:display();
```

```
break;
case 3:insert_at_begin();
break;
case 4:insert_at_end();
break;
case 5:insert_after_position();
break;
case 6:insert_before_position();
break;
case 7:delete_first_node();
break;
case 8:delete_last_node();
break;
case 9:delete_specific_node();
break;
case 10:getlength();
break;
case 11:search_node();
break;
case 12:exit(0);
break;
default:printf("enter valid input");
break;
}
printf("\n continue(Y/N):- ");
fflush(stdin);
scanf("%c",&ch);
}while(ch=='Y' || ch=='y');
getch();
}
```

```
enter your choice:- 1
enter data:- 20

 continue(Y/N):- y

enter your choice:- 3
enter data:- 50
data in the list are as follow
50      10      20
 continue(Y/N):- y

enter your choice:- 5
after which position you want to insert:- 1
enter data:- 70
data in the list are as follow
50      70      10      20
```

PRACTICAL – 12

```
#include <stdio.h>
void swap(int *xp, int *yp)
{
int temp = *xp;
*xp = *yp;
*yp = temp;}
void bubbleSort(int arr[], int n)
{
int i, j;
for (i = 0; i < n-1; i++)
for (j = 0; j < n-i-1; j++)
if (arr[j] > arr[j+1])
swap(&arr[j], &arr[j+1]);}
void printArray(int arr[], int size)
{
int i;
for (i=0; i < size; i++)
printf("%d ", arr[i]);
printf("\n");}
int main()
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
int arr[] = {64, 34, 25, 12, 22, 11, 90};
int n = sizeof(arr)/sizeof(arr[0]);
bubbleSort(arr, n);
printf("Sorted array: \n");
printArray(arr, n);
return 0;}
```

```
Name: Yash Tripathi
En.no.: 210410107140
Sorted array:
11 12 22 25 34 64 90
```

PRACTICAL – 13

```
#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(){
   printf("Name: Yash Tripathi\n");
   printf("En.no.: 210410107140\n");
   int i, count, number[25];
   printf("How many elements are u 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;
}
```

```
Name: Yash Tripathi
En.no.: 210410107140
How many elements are u going to enter?: 4
Enter 4 elements: 8
3
5
9
Order of Sorted elements:  3 5 8 9
```

PRACTICAL – 14

```
#include <stdio.h>
#include <stdlib.h>
void merge(int arr[], int l,
           int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
```

```
Name: Yash Tripathi
En.no.: 210410107140
Given array is
12 11 13 5 6 7

Sorted array is
5 6 7 11 12 13
```

```
        }
}
void mergeSort(int arr[],
                        int l, int r)
{
        if (l < r)
        {
                int m = l + (r - l) / 2;
                mergeSort(arr, l, m);
                mergeSort(arr, m + 1, r);

                merge(arr, l, m, r);
        }
}
void printArray(int A[], int size)
{
        int i;
        for (i = 0; i < size; i++)
                printf("%d ", A[i]);
        printf("\n");
}
int main()
{
   printf("Name: Yash Tripathi\n");
   printf("En.no.: 210410107140\n");
        int arr[] = {12, 11, 13, 5, 6, 7};
        int arr_size = sizeof(arr) / sizeof(arr[0]);

        printf("Given array is \n");
        printArray(arr, arr_size);

        mergeSort(arr, 0, arr_size - 1);

        printf("\nSorted array is \n");
        printArray(arr, arr_size);
        return 0;
}
```

PRACTICAL – 15

```
#include <stdio.h>
int main()
{
printf("Name: Yash Tripathi\n");
printf("En.no.: 210410107140\n");
  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;
}
```

```
Name: Yash Tripathi
En.no.: 210410107140
Enter number of elements
4
Enter 4 integers
6
34
8
99
Enter value to find
34
34 found at location 2.
```