ASSIGNMENT

G.LIKHITA
AP19110010489

1) Program to insert and delete an element at the nth and kth position in a linked list where n and k is taken from user.

Ans)

```
#include<stdio.h>
#include<stdlib.h>
{
int d;
struct node* next;
}
display(struct node* head)
{
if(head==NULL)
{
printf("NULL\n");
}
else
{
printf("%d\n", head->d);
display(head->next);
}
}
del(struct node * ~~dat~~ before_del)
{
struct node * temp;
temp = before_del->next;
before_del->next = temp->next,
free(temp);
}
struct node * front(struct node *head, int value)
{
  struct node * t;
  t = malloc(size of (struct node));
  t-> ~~dat~~ d = value
  t->next = head;
```

```
    return (t);
}
end (struct node * head; int value)
{
struct node * t, * m;
t = malloc (size of struct node));
 t -> d = value;
 t -> next = NULL;
 m = head;
 while (m -> next != NULL)
 {
   m = m -> next;
 }
  m -> next = t;
}
 after (struct node * z, int value)
 {
   if (z -> next != NULL)
 {
 struct node * t;
 t = malloc (size of (struct node));
 t -> d = value;
 t -> next = z -> next;
 z -> next = t;
 }
 else
 {
   printf ("END Function to be used to insert at the
                                          end \n");
 }
 }
 int main ()
 {
 struct node *prev, * head, * t;
 int z, i;
 printf (" Numbes of elements are :")
 scanf ("%d", & z);
```

```
head = NULL;
for (i=0; i<z; i++)
{
  t = malloc (size of (struct node));
  scanf ("%d", &t -> d);
  t -> next = NULL;
  if (head == NULL)
  head = t;
  else
  prev -> next = t;
    prev = t;
}
head = front (head, 20)
end (head, 40);
after (head -> next -> next, 60);
def (head -> next);
del (head -> next -> next);
display (head);
return 0;
}
```

OUTPUT :

Number of element are : 5

1
2
3
4
5
20
1
60
4
5
40
NULL

2) Construct a new linked list by merging alternate nodes of two lists for example in list 1 we have {1, 2, 3} and in list 2 we have {4, 5, 6} in the new list we should have {1, 4, 2, 5, 3, 6}

Ans)

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
int d;
struct Node * next;
};
void push(struct Node ** head_ref, new_d)
{
struct Node * new_node = (struct Node *) malloc (sizeof (struct Node));
    new_node -> d = new_d;
    new_node -> next = (*head_ref);
     (*head_ref) = new_node)
}
void print list (struct Node *head)
{
struct Node * temp = head;
while (temp != NULL)
{
 printf ("%d", temp -> d);
 temp = temp -> next;
}
printf ("\n");
}
void merge (struct Node * p, struct Node *q)
{
struct Node * p_current = p; *q_current = *q;
struct Node * p_next, *q_next;
while (p_current != NULL && q_current != NULL)
```

```
    t->next = t->current->next;
    q->next = q->current->next;
    q->current->next = t->next;
    t->current->next = q->current;
    t->current = t->next;
    q->current = q->next;
}
*q = q->current;
}
int main()
{
struct Node *t = NULL; *q = NULL;
push(&t, 0);
push(&t, 1);
push(&t, 4);
printf("First linked list\n");
printlist(t);
push(&q, 2);
push(&q, 3);
push(&q, 5);
printf("Second linked list\n");
printlist(q);
merge(t, &q);
printf("New linked list\n");
printlist(t);
return 0;
}
```

OUTPUT :

First linked list

0 1 4

Second linked list

2 3 5

New linked list

0 2 1 3 4 5

3) Find all the elements in the Stack whose sum is equal to k (where k is given from the user)

A)

```
#include <stdio.h>
int top=-1;
int a;
char stack[1000];
void push(int a);
char pop();
int main()
{
int i,n,x,l,m,sum=0,count=1;
printf("Enter the number of elements:");
scanf("%d",&n);
for(i=0;i<n;i++){
printf("Enter the next element :");
scanf("%d",&x);
push(x);
}
printf("Enter the sum");
scanf("%d",&m);
for(i=0;i<n;i++)
{
l=pop();
sum+=l;
count+=1;
if(sum==m){
for(int j=0;j<count;j++)
printf("%d",stack[j]);
p=1;
break;
}
push(l);
}
if(p!=1)
```

```
printf ("The elements don't add up to the sum");
}
void push(int a)
{
if (top==999)
{
printf("\n Stack is FULL");
return;
}
top= top+1;
stack[top]= a;
}
char pop()
{
if (stack[top] ==-1)
{
printf("\n Stack is Empty");
return 0;
}
a= stack[top];
top = top -1;
    return x;
}
```

~~OUTPUT~~: Output :

```
Enter the number of element : 8
Enter the next element : 26
Enter the next element : 67
Enter the next element : 56
Enter the next element : 78
Enter the next element : 99
Enter the next element : 43
Enter the next element : 76
Enter the next element : 32
Enter the sum : 400
The element don't add upto the sum
```

4th Program:

Write a program to print the elements in a queue

i) in reverse order

ii) in alternate order

A)

```
#include <stdio.h>
#define SIZE 10
void insert(int);
void delete();
int queue[100], a=-1, b=-1;
void main(){
  int value, choice;
  while(1){
    printf("\n\n***MENU***\n");
    printf("1. Insertion\n 2. Deletion\n 3. Print Reverse\n
            4. Print Alternate\n 5. Exit");
    printf("\n Enter your choice :");
    scanf("%d", &choice);
    switch(choice){
case 1: printf("Enter the value :");
scanf("%d", &value);
insert(value);
break;
case 2: delet()
break;
case 3:
    printf("The reversed queue is:");
    for(int i=SIZE; i>=0; i--)
{
    if(queue[i]==0)
    continue;
```

```
        printf("%d", queue[i]);
    }
    break;
case 5: exit(0);
default: printf("\n Wrong selection !!! Try again !!! ");
    }
}
void insert(int value){
    if((a==0 && b==SIZE-1) || a==b+1)
        printf("\n Queue is Full !!! Insertion is not possible !!! ");
    else {
        if(a==-1)
a=0;
    b=(b+1)%SIZE;
    queue[b]=value;
    printf("\n Insertion success !!! ");
}
}
void delete(){
  if(a==-1)
      printf("\n Queue is empty !!! Deletion is not possible !!! ");
  else {
      printf("\n Deleted: %d", queue[a]);
      a=(a+1)%SIZE;
      if(a==b)
a=b=-1;
}
}
```

5) i) How array is different from the linked list

ii) Write a program to add the first element of one list to another list

Array	Linked list
1) size of an array is fixed	size of a list is not fixed
2) It occupies less memory than a linked list for the same number of elements	It occupies more memory
3) Deleting an element from an array is not possible	Deleting an element is possible
4) Insertion and deletion take more time	Insertion and deletion process take less time

ii)

```
#include<stdio.h>
#include <stdlib.h>
struct Node
{
 int d;
 struct Node * next;
};
void push (struct Node** head_ref, int, new_d)
{
struct Node * new_node = (struct Node*) malloc (size of
                                                (struct Node));
new_node -> data = new_data;
new_node -> next = (*head_ref);
(*head_ref) = new_node;
}
```

```
void printList (struct Node *head)
{
    struct Node * temp=head;
    while (temp != NULL)
    {
     printf ("%d ", temp -> d);
        temp = temp -> next;
    }
    printf ("\n");
}
void merge (struct Node *a, struct Node **b)
{
    struct Node *a_current= a, *b_current = *b;
    struct Node *a_next, *b_next;
    while (a_current != NULL && b_current != NULL)
    a_next = a_current -> next;
    b_next = b_current -> next;
    b_current -> next = b_current;
    a_current = a_next;
    b_current = b_next;
    }
    *b= b_current;
}
int main()
{
    struct Node *a =NULL, *b= NULL;
    push (&a, 1);
    push (&b, 3);
    push (&b, 5);
    printf ("First Linked list \n");
    printf list (a);
```

```
push(&b, 1);
push (&b, 4);
push (&b, 8);
push (&b, 10);
push(&b, 3);
printf("Second linked list \n");
print list (b);
merge (a, &b);
printf ("New First linked list \n");
print list (a);
printf ("New second linked list \n");
print list (b);
```

OUTPUT :

First linked list

1 3 5

Second linked list

1 4 8 10 3

New First linked list

1 1 3 4 5 8

New Second linked list

10 3