K.UMA PRANAVI
AP19110010375
CSE-F

1. Write a program to insert and delete an element of the nth and kth position in a linked list. where n and k is taken from user.

sol:-

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
 struct node *next;
};
struct node *curr,*temp;
void input (struct node*s)
void delete (struct node*s)
void main (void)
{
 struct node *s;
 int n;
 s=null;
do
{
printf ("Enter the element to insert ;\n");
printf("2. Delete\n");
printf("3. Exit\n");
printf("Enter the choice:");
 scanf("%d",&n);
switch(n)
{
case 1: input(s);
        break;
Case 2: delete(s);
        break;
 } while (n:=3)
```

```
}
void input (struct node *z)
{
  int pos, c=1;
   curr=z;
 printf("Enter the element to be inserted : ");
 scanf("%d", &pos);
 while (curr->next!=null)
 {
  c++;
 if (c==pos)
{
temp=(struct *) malloc (size of (struct node));
printf("Enter the numbers:");
scanf("%d", &temp->n);
 temp->next=curr->next;
  curr->next=temp;
  break;
 }
}
}
void delete (struct node *z)
{
 void delete (struct node *z)
 {
  int pos, c=1;
  curr=z;
 printf("enter the element to be deleted:");
 scanf("%d", &pos);
 while(curr->next!=null)
 {
  c++;
 if(c==poss)
 {
```

```
temp = (struct node*) malloc (size of (struct node));
printf ("Enter the numbers: ");
scanf ("%d", &temp->n);
temp->next = curr->next;
curr->next = temp;
break;
}
}
}
void delete (struct node *z)
{
int pos, c=1;
curr = 2;
printf ("enter the element to be deleted:");
scanf ("%d", &pos);
while (curr->next != NULL)
{
c++;
if (c==pos)
{
temp = curr->next;
curr->next = curr->next->next;
free (temp)
}
curr = curr->next;
}
void merge (struct node*p, struct node*q)
{
struct node* p-curr = p, *q -curr = *q;
struct node* p-next, q-next;
while (p-curr = Null && q-curr != NULL)
{
```

```
    p->next = p->curr->next;
    q->next = q->curr->next;
    q->curr->next = p->next;
    p->curr->next = q->curr;
    p->curr = p->next;
    q->curr = q->next;
  }
  *q = q->curr
}
int main()
{
  struct node *p = NULL, *q = NULL;
   push(*p,1);
   push(*p,2);
   push(*p,3);
   printf("First linked list:\n");
   printlist(R);
   push(*q,4);
   push(*q,5);
   push(*q,6);
   printf("second linked list:\n");
   printlist(q);
   merge(p,*q);
   printf("modified first linked list=\n");
   printlist(p);
   printf("modified second linked list=\n");
   printlist(q);
   return 0;
}
```

2. Construct a new linked list by merging alternatives notes 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}

```
#include <stdio.h>
# include< stdlib.h>
#include < assert.h>
struct node
{ int data;
  struct node * next;
};
void move node(struct node **x ; struct node **y);
struct node * sorted merge (struct node*a, struct node*b)
{
  struct node dummy;
  struct node *tail = &dummy;
  dummy.next = NULL;
  while (1)
  {
    if(a==NULL)
  {
    *y = newnode -> next;
    newnode -> next = *x;
    *x = newnode;
  }
void push(struct node ** head_ref, int new_data)
{
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 print list (struct node *node)
{
while (node != NULL)
{
printf("%d", node->data);
node = node->next;
}
}
    tail->next=b;
    break;
}
else if (b==NULL)
{
~~else if (b~~ tail->next = a;
    break;
}
if (a->data <= b->data)
{
move node {&(tail)->next, &a};
}
else
{
move node *~~newnode = *y~~; (&tail)->next, &b);
}
tail = tail->next;
}
return (dummy.next);
}
void movenode (struct node** x, struct node **y)
{
struct node* newnode = *y;
assert(newnode != NULL);
```

```
int main()
{
struct node *res = null;
struct node *a = Null;
struct node *b = Null;
Push (&a,2);
Push (&a,3);
Push (&b,4);
Push (&b,5);
Push (&b,6);
res = sorted merge (a,b);
printf("merge linked list is :\n");
print list (res);
return 0;
}
```

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

```
#include <stdio.h>
int s1[10], top1=-1, s2[10], top2=-1;
int s1 empty ()
{
 if (top1==-1)
  return 1;
 else
   return0;
}
 int s1 top()
{
 return s1[top1];
}
 int s1 pop()
{
```

```
return s1[top1];
}
int s1pop()
{
 top1--;
}
int s1push(int x)
{
 s1[++top1] = x;
}
 int s2empty()
{
 if (top2 == -1)
 return 1;
 else
 return 0;
}
int s2top()
{
 return s2[top2];
} int s2pop()
{
 top2--;
}
int s2push(int x)
{
 s2[++top2] = x;
}
 int sum(int k)
{ int x;
```

```
while(s1empty( ) != 1)
{
 x=s1 top( );
 s1 pop( );
while(s1empty( ) != 1)
{
 if (x+s1 top( ) = k)
{
 printf("%d, %d \n", x, s1top( ));
}
  s2push(s1top( ));
   s1pop( );
}
 while(s2empty( ) != 1)
 {
  s1push(s2topt( ));
   s2 pop( );
  }
}
  int main( )
{ int n, i, e, k;
 printf("Enter the no of elements of stack: \n");
  scanf("%d", &n);
 for (i=0; i<n; i++)
 {
  scanf ("%d", &e);
  s1 push(e);
 } printf("enter the value of constant sum: \n");
   scanf("%d", &k);
```

```
printf("The combinations whose sum is equal to k is :\n");
    sum(k);
}
```

4. Write a program to print the elements in a queue
(i) in reverse order
(ii) inalternate order.

sol:

```
(i) #include <stdio.h>
#include <stack.h>
#include "qq.h"
int main()
{
 int n, arr[20], i, j=0;
 struct stack s;
 initstack(&s);
 printf("Enter no");
 scanf("%d", &n);
 for(i=0; i<n, i++)
  {
    printf("Enter values : ");
    scanf("%d", &arr[i]);
}
for(i=0; i<n; i++)
{
 insert(arr[i]);
}
 while(j!=n)
{
push(&s, del());
j++;
```

```
}
printf("Reverse is ");
while (stop != -1)
{
printf("%d", pop(&s));
}
printf("\n");
return 0;
}
```

(ii)

```
#include <stdio.h>
#include <stdlib.h>
struct node {
int data;
struct node *next;
}
void print nodes (struct node * head)
{ int count = 0;
while (head != NULL)
{ if (count % 2 == 0)
{ printf("%d ", head->data);
}
count++;
head = head -> next;
}
}
void push (struct node ** head-ref, int new_data)
{
```

```
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;
}
int main()
{
struct node* head = NULL;
 push(&head, 12);
 push (&head, 29);
 push(&head, 11);
push (&head, 23);
push (&head, 8);
print node(head);
return 0;
}
```

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

sol: The major difference b/n array and linked lists regards to their structure. Arrays are index based data structure where each element associated with an index. On the other hand, linked list relies on the reference to previous and next element.

(ii) Write a program to add the first element of one list to another list If example we have {1,2,3} in list1 and {4,5,6} in list2 we have to get {4,1,2,3} as output for list 1 and

and {5,6} for list2.

sol:

```
#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 (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 -> data);
   temp = temp -> next;
 }
 printf("\n");
}
```