Assignment

D. Lavanya Sujana
AP19110010470
CSE-H

1. Write a program to insert and delete an element at the n^{th} and k^{th} position in a linked list where n and k is taken from user.

Ans:

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   struct node *next;
};
struct node *curr, *temp;
void input (struct node*)
void delete (struct node*)
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(5);
            break;
    case 2: delete(5);
            break;
}
while(n!=5)
}
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=(structnode*)malloc(sizeof
                              (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 = z;
  printf("\nEnter the element to be
                                delete:");
  scanf("%d", &pos);
  while (curr->next != Null)
  {
    c++;
    if (c==pos)
    {
      temp= curr->next;
      free(temp)
    }
    curr = curr=curr->next;
}
void merge(struct node *p, structnode *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_next;
        p_curr = p_next;
        q_curr = q->next;
    }
int main ( )
{
    struct node *p = Null, *q = Null;
     push (&p, 1);
     push (&p, 2);
     push (&p, 3);
     printf("first linked list: \n");
     print list (R);
     push (&q, 4);
     push (&q, 5);
     push (&q, 6);
      printf ("second linked list: \n");
      print list (q);
       merge (p, &q);
       print f (" modified first linkedlist
                                    = \n");
       print list (p);
       print f ("modified second linked list
                                    = \n");
       print list (q);
       return 0;
    }
```

1. Construct a new linked list by merging alternatives 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}.

```
#include <stdio.h>
#include <stdlib.h>
#include <assertin>
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 = new node->next;
        newnode->next = *x;
        *x = new node
      }
}
void push(struct node** head_ref, int new_data)
```

```
{
  struct node *new_node = (struct node *)malloc(size
                                of (struct node));
  new_data -> data = new_data;
  new_data -> next = (*head_ref);
  (*head_ref = new_node;
}
void print list (struct node * node)
{
   while(node) = Null)
   {
      print f ("%d", node -> data);
      node = node -> next;
   }
}
tail ->next = b
  break;
}
else if (b = Null)
{
   tail -> next = a;
    break;
}
if(a -> data <= b -> data)
{
   move node { &(tail) -> next), &a);
}
else
{
   move node (&(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, 1);
  push(&a, 2);
  push(&a, 3);
  push(&b, 4);
  push(&b, 5);
  push(&b, 6);
  res = SortedMerge(a, b);
  printf("Merge Linked List is: \n");
  printList(res);
  return 0;
}
```

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

Ans:

```
#include<stdio.h>
int s1[10], top1 = -1, s2[10], top2 = -1;
```

```
int S1 empty()
{
  if (top1 == -1)
    return 1;
  else
    return 0;
}
int S1 pop()
{
  top1--;
}
int S1 push(int x)
{
  S1[++top1] = x;
}
int S2 empty()
{
  if (top == -1)
    return 1;
  else
    return 0;
}
int S2 top()
{
  return S2[top2];
}
int S2 pop()
{
  top2--;
}
int S2 push(int x)
{
```

```
    s2[++top2]=x;
}
int sum(int k)
{
    int x;
    while(s1.empty()!=1)
    {
        x = s1.top()
        s1.pop();
        while(s1.empty()!=1)
        {
            if(x+s1.top()==k)
            {
                printf("%d,%d\n", x, s1.top());
            }
            s2.push(s1.top());
            s1.pop();
        }
        while(s2.empty()!=1)
        {
            s1.push(s2.top());
            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:");
scanf("%d",&k);
printf("The combination whose sum is equal to k is:\n");
sum(k);
}
```

4. Write a program to print the element in a queue
   i. in reverse order
   ii. in alternate order

```
#include<stdio.h>
#include"stack.h"
#include"QQ.h"
int main()
{
    int n,arr[20],i,j=0;
    struct stack s;
    int stack(<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++;
    }
    print("Reverse is ");
    while(stop1 != -1)
    {
        printf("%d", pod(&s));
    }
    print f("\n");
    return 0;
}
```

ii)

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *next;
};
void print node(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
                            (sizeof(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, 11);
    push(&head, 10);
    push(&head, 6);
    push(&head, 23);
    printnode(head);
    return 0;
}
```

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 of example we have {1,2,3} in list 1 and {4,5,6} in list 2 we have to get {4,1,2,3} as output for list 1 and {5,6} for list 2.

Ans:- i) The major difference between array and linked lists regarding to their structure, Arrays are index data structure where each element associated with an index. On the other hand, linked list relies on reference to the previous and next element.

ii)
```
#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 print list ( struct node *head)
{
```

```
struct node * temp = head;
while (temp != Null)
{
    print f("%d", temp -> data);
    temp = temp -> next;
}
print f("\n");
}
```