R. Vamsi Krishna
AP19110010500

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.

Sol:
```
#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(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 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 delete");
  Scanf("%d", &pos);
  while(curr->next != NULL)
{
  c++;
  if(c==pos)
{
  temp = current->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_curV = Null && q_curV != Null)
{
  P_next = P_curV -> next;
  q_next = P_curV -> next;
  q_curV -> next = P_next;
  P_curV -> next = q_curV;
  P_curV = P_next;
  q_curV = q_next;
}
*q = q_curV
}
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");
Printf list(q);
merge(P,&q);
Printf("Modified first linked list=\n");
Print list(P);
Printf("Modified Second linked list=\n");
Print List(q);
return 0;
}
```

2) Construct a new linked list by merging alternative ~~with~~ notes of two lists for ex. 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}

A)
```
# 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 = new node → next;
     new node → 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_node → data = new-data;
 new-node → next = (* head-ref);
 (* head-ref) = new-node;
}
Void Point list (struct node * node)
```

```
{
while (node != NULL)
{
Printf ("%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(&a,4);
  Push(&a,5);
  Push(&a,6);
 res = Sorted merge(a,b);
 Printf("merge linked list is: \n");
 Printf(
  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)

A)
```
# 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 top()
{ return S1[top1];
}
int S1 Pop()
{ top1--;
}
int S1 Push(int x)
{ S1[++top1] = x;
}
int S2 empty()
{
  if (top2 == -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("%od", &n);
    for(i=0; i<n; i++)
    {
        Scanf("%od", &e);
        S1.Push(e);
    }
Printf("enter the value of Constant Sum:\n");
Scanf("%od", &k);
Printf("The Combinations whose Sum is equal
        to k is:\n");
    Sum(k);
}
```

4) Write a Program to Print the elements in a q
(i) in reverse order
(ii) in alternate order

A)
(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);
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(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, 29);
  Push(&head, 11);
  Push(&head, 23);
  Push(&head, 8);
  Print node(head);
  return 0;
}
```

5) (i) How array is different from 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 list2 we have to get {4,1,2,3} as output for list 1 and {5,6} for list 2

A) (i) The major difference b/w assay and linked lists regards to their structure. Assays are index based 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
                        (Size of(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)
  {
    Printf("%d", temp->data);
    temp = temp->next;
  }
  Printf ("\n");
}
```