-AP19110010599
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.

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)
{
 stud 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 inpos( 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 * x)
{
  int pos, c=1;
  curr = x;
  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_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");
Print list (p);
Push (&q, 4);
Push(&q, 5);
Push(&q, 6);
Printf("second linked list: \n");
Print 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;
}
```

Output:-

First linked list : 1 2 3
Second linked list : 4 5 6

(2) 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}

Sol

```
#include <stdio.h>
#include <stdlib.h>
#include<assert.h>
struct node
{
  int data;
  struct node * next;
};
void move node(struct node ** a, struct node **b);
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)
 {
```

```
        tail -> next = b;
        break;
    }
    else if (b == NULL)
    {
        tail -> next = a;
        break;
    }
    if (a -> data <= b -> data)
    {
        MoveNode (&(tail -> next), &a);
    }
    else
    {
        MoveNode (&(tail -> next), &b);
    }
    tail = tail -> next;
}
return (dummy.next);
}
void MoveNode (struct node **x, struct node **y)
{
    struct node* newNode = *y;
    assert (newNode != 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
                          (sizeof(struct node));
  new_node -> data = new_data;
  new_node -> next = (*head_ref);
  (*head_ref) = new_node;
}
void printList (struct node * node)
{
  while (node != NULL)
  {
    Printf("%d", node -> data);
    node = node -> next;
  }
}
```

```
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);
  Pusn(&b, 5);
  Push (&b, 6);
  res = sorted merge (a b);
  Printf (" merge linked list is: \n");
  Print list (res);
  return 0;
}
```

Output :-

Merge linked list is:

1 4 2 5 3 6

(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 s1empty()
{
  if (top1 == -1)
      return 1;
  else
     return 0;
}
int s1top()
{
  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 s2 push(int x)
{
   s2[++top2] = x;
}
  int sum (int k)
{
    int x;
    while (s1 empty() != 1)
    {
        x = s1top();
        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);
    s1push(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);
}
```

Output:-

```
Enter the no. of elements in stack: 5
1
2
3
4
5
enter the value of constant sum: 5
The combination whose sum is equal to k is:
(4,1)
(3,2)
```

(4) Write a program to print the elemts in a queue.
i) in reverse order.
ii) in alternate 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++)
  {
    Print("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 (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;
}
```

(i) output :-

```
Enter no : 3
Enter values : 1
2
3
Reverse
3
2
1
```

(ii) output :-

```
head -data
12  19  11  23  8

head
alternate

12  11  8
```

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 for 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.

Sol (i) The major difference b/w 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 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 printlist(struct node *head)
{
  struct node *temp = head;
  while(temp != NULL)
  {
    printf("%d", temp->data);
    temp = temp->next;
  }
  Printf("\n");
}
```

Output:-

data in First linked list: 2 3 4 5
data in Second linked list: 6 7 8 9
new_data = 2 6 7 8 9 , 3 4 5