M. Roop Sagar
AP19110010376
CSE-f

① Write a program to insert and delete an element of 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 * curv, *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;");
print f (" 2. Delete \n");
print f (" 3. Exit \n");
print f (" Enter the choice :");
```

```
scanf ("%d", &n);
    switch (n)
    {
    case 1: input (s);
            break;
    case 2: input (s)
            break;
    } while (n: = 3)
}
void input (struct node * x)
{
int   pos , c=1
curr I x;
printf ("Enter the element to be inserted:");
scanf ("%d", &pos);
    while (cur → 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 = z;
    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 link: \n")
printf list (R));
Push (&q,4)
push (&q,5)
push (&q,6)
printf ("second linked list :\n");
printf List (q);
merge (P,dq);
 printf ("modified first linked list = \n");
Print List (p);
print f ("modified second linked list = \n")
print list (q);
return 0;
}
```

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 <assertin>
 struct node
{
  int data;
  struct node *next;
};
void move node (struct node ** x; struct node * * y);
 struct node *  sorted merge (struct node * o, struct
                                node * b)
{
 struct node dummy
 struct node * toil = & dummy;
 dummy . next = Null;
 while (1)
{
  if (a == Null)
{
```

```
    *y = newnode ⟶ 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;
  }
  elseif (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 move node * (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);
  print f ("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)

Sol
```
# include <stdio.h>
int S1[10], top1 = -1, S2[10], top2 = -1
int S1 empty ()
{
  if (top1 == -1)
     return 0;
}
int S1 top ()
{
  return S1[top1]
}
int S1 pot ()
{
 int top1--;
}
int S1 push (int x)
{
   S1[++top1] = x
}
int S2 empty ()
{
```

```
    if (top2 == -1)
        return ;
    else
        return 0,
}
int S2 top( )
{
 return S2 [top 2];
}
  int S2 pop( )
 {
  top 2 -- ;
 }
 int S2 push (int x)
 {
  S2 [++ top 2] = 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 ; kn; i++)
{
  scanf ("%d", &e);
  slpush(e);
}
printf ("enters 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 as;
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 (i!=n)
{
  push (ds, 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 noder new_node = (struct node *)
                            mollo c (size of (strucf 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);
    printf node (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.

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
                                    (size of (struct node));
    new_node -> data = new_data;
    new_node -> next = (&head_ref);
      (*head_ref) = new_node;
}
  void printf list (Struct node * head)
{
  struct node * temp = head;
  while (temp != Null)
  {
    printf ("%d", temp -> data);
    temp = temp -> next;
  }
  printf ("\n");
}
```