P. Jayanth kumar
AP19110010332

1) Write a program to insert and delete an element at the n^{th} & k^{th} position in a linked list when 'n' & k are taken from user

Code

```
# 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);
             breack;
     } while (n != 3)
}
void input (struct node * z)
{
  int pos, c = 1
  curr = z;
  Printf ("Enter the elements to be inserted: ");
  Scanf ("%d", &pos);
  while (curr -next != Null)
   { c++;
     If (c==pos)
     {
        temp= (struct node*) malloc (sizeof (structnode));
        Printf ("Enter the number : ");
        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 list:\n");
     Print list(p);
      Push(&q,4);
       Push(&q,5);
        Push(&q,6);
     Printf("second linked list:\n");
     Printfist(p);
      Printf("modified second linked list=\n");
          Print list(q);
           return 0;
     }
```

Consult a new linked list by merging alternative nodes of two lists for Examples in list -1 we have {1,2,3} & in list 2 we have {4,5,6} in the new list we should have {1,4,2,5,3,6}

Code 1:

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
struct node
{
  int data;
   struct node * next;
};
void movenode(struct node ** x, struct node**y);
struct node* sortedmerge(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 _ eref);
  (* 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 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
 struct Node *
   Push (&a, 1);
    Push (&a, 2);
     Push (&a, 3);
    Push (&a, 4);
    Push (&a, 5);
    Push (&a, 6);
  res = sorted merge (a, b);
  Printf ("merged linked list is : \n");
  Printbe;
    Print list (res);
   return 0;
}
```

③ find all the elements in the stack whose sum is Equal to k

```
#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;
```

```
else
    return 0;
}
int s2 topC)
{
  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: \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) in alternate order.

(i)
```
#include <stdio.h>
#include "stack.h"
#include "Q.Q.h"
```

(ii)

```
#include <stdio.h>
#include <stdlib.h>
struct node{
int data;
struct Node * next;
}
Void printnodes (struct Node *head)
{
    int count=0
    while (head! = Null) {
    If (Count %. 2 == 0){
      Printf ("%.d " head → data);
      }
      Count ++;
       head = head → next;
       }
    }
Void push (Strud 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 ()
{
    int n, arr[20], i, j = 0;
    struct stack s;
    init 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++;
    }
    Printf ("reverse is");
    while (stop != -1)
    {
      Printf ("%d", Pop(& s));
    }
    Printf ("\n");
    return 0;
}
```

```
int main ()
{
  struct node * head = Null;

      Push (& head , 12);
       Push (& head , 29);
      Push ( & head , 11)
       Push (& head , 23)
       Push (& head , 8);
       Point 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 lists and {5,6} for list 2.

(i) The Major difference b/w Array and linked lists regards to their structure. Arrays are index based data structure where each elements associated with an index. On the other hands linked list relies on reference to the previous and next element.

(ii) #include <stdio.h>
#include <stdlib.h>

```
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");
  }
```