Assignment - 4

Yogitha Goli
AP19110010320
CSE-F

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

```
#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 for insert; \n");
Printf ("2. Delete \n");
Printf ("3. Exit \n")
Printf ("Ente 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 *) malloc (size of (structnode));
Printf ("Enter the numbers: ");
 Scanf ("%d", & temp → n);
   temp → next = curr → next;
       curr → next = temp;
       break;
     }
   }
 }
  void delete (struct node * z)
 {
   void delete (struct node * z)
 {
   int pos , c = 1;
    curr = z;
```

```
Printf("Enter the element to be deleted:");
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 deleted:");
    Scanf("%d", &pos);
     while (curr -> next != null)
   {
     c++
      if(c == pos)
   {
     temp = curr -> 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 (K);
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;
}
```

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}

```
#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;
   newnode → next = *x;
    *x = new node;
}
 void push ( struct node ** head - ref, int new - data)
{
 Struct node * new - node = (struct node *) malloc ( sizeof
 ( struct data)
   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, 2);
 Push(&b, 3);
 Push(&b, 4);
 Push(&b, 5);
 Push(&b, 6);
```

```
res = sorted merge (a, b);
Printf ("merge linked list is : \n");
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 SI top()
{
  return SI (top1);
}
  int S1 Pop()
{
  return S1(top1);
}
  int S1 pop()
{
  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, s2.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 for print the elements in a queue

(i) inverse order

(ii) inalternate order.

(i)

```
#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, de[c]);
  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);
  printf("%d", printnode(head));
  return 0;
}
```

5)(i) How array is different from the linked list

Sol: The major difference between array and linked list 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 the reference for previous and next element.

(ii) Write a program for add the first element of one list for another list if example we have {1, 2, 3} in list 1 and {4, 5, 6} in list 2 we have for get {4, 1, 2, 3} as output for list 1 and {5, 6} for list 2

```
#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)
    {
      Printf ("%d ", temp -> data);
       temp = temp -> next;
    }
      Printf ("\n");
  }
```