Assignment-4

K.Manasa
AP19110010343
CSE-H

1) write a Program to insert and delete an element at the nth and kth Position in a linked list when n and k is taken from user.

```
#include <stdio.h>
#include <stlib.h>
struct Node
{
    int data;
    struct Node* next;
};
struct Node* delete(struct Node* head, intn);
struct Node* insert (struct Node* head, int n);
struct Node* create_list ();
void display (struct Node*);
void main()
{
    int k;
    struct Node* head;
```

```
        head = create_list();                    ②
        display (head);
        Printf ("enter the index where you want to ente
        scanf ("%d", &k);
        head = insert (head, k);
        display (head);
        head = delete (head, 3);
        display (head);
}
void display (struct Node* head)
{
    struct Node* p;
    for (p= head; p! =NULL; p = p-> next)
    {
        Printf ("\n Node data %d", p->data);
    }
    Printf ("\n");
}
struct Node* create_list()
{
    int k, n;
    struct Node* p, * head;
    Printf ("\n How many elements to enter");
```

```
scanf ("%d", &n);
for (k=0; k<n; k++)
{
        if (k==0)
        {
         Head = (struct Node*)malloc(sizeof(
                                       (structNode));
         P= Head;
        }
        else
        {
         P -> next =(struct Node*)malloc(size of(
                                        (struct(Node)
         P = P-> next;                               ;
       }
   printf ("\n Enter an %dth element", k);
   scanf ("%d", &p -> data);
}
P-> next = NULL;
return (Head);
}
struct Node* insert (struct Node * head, int n)
{
   int i=0;
   struct Node*p, * temp;
```

```
        p = head;
        temp =(struct Node*)malloc(sizeof(struct
        while (i! = n)                       Node));
        {
            p = p->next;
            i++;
            if (i= =n)
            {
            printf("enter the element that you want
                       to enter");
            scanf ("%d", & temp->data);
            temp -> next = p->next;
            p-> next= temp;
        }
        }
        return (head);
}
struct Node * delete (struct Node * head, intn)
{
    int i=0;
    Struct Node * p, * temp;
    p= head;
    while (i! =n-1)
```

```
        {
            P= P -> next
            i++ ;
            if (i= =n-1)
            {
              P-> next =(P-> next) ->next;
            }
        }
        return (head);
}
```

OUTPUT

```
Enter the 1th element 2
Enter the 2nd element 3
Enter the 3th element 4
Node data 1
Node data 2
Node data 3
Node data 4
Enter the index where you want to enter 1
Enter the element where you want to enter 4
```

```
Node data 1
Node data 2
Node data 44
Node data 3
Node data 4

Node data 1
Node data 2
Node data 44
Node data 4
```

2.) construct a new linked list by merging alternate nodes of two lists for example in list1 we have {1,2,3} and in list2 we have {4,5,6} in the new list we should have {1,4,2,5,3,6}

```
# include <stdio.h>
# include <stlib.h>
struct Node
{
        int data;
        struct Node* next;
};
void Print List (struct Node* head)
{
            struct Node* ptr = head;
            while (ptr)
            {
                Printf ("%d → "; ptr → data);
                ptr = ptr → next;
            }
            Printf ("NULL\n");
}
```

```
void push (struct Node* head, int data)
{
        struct Node* newNode =(struct Node*) Malloc
                                    (size of(struct Node));
        Newnode → data = data;
        newnode → Next = head;
        * head = newNode ;
}
struct Node* shuffle 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;
            }
            else
            {
                    tail → next = a;
                    tail = a;
                    a = a → next;
                    tail → next=b;
                    tail = b',
                    b= b → next ;
              }
        }
}       return dummy. next;

int main (void)
{
            int keys[] = { 1, 2, 3, 4, 5, 6, 7 };
            intn = size of (Keys) / size of( keys[0]);
            struct Node* a= NULL , * b= NULL;
```

```
for (int i = n-1; i>=0; i = i-2)
        Push (&a, keys [i]);
for (int i = n-2; i>=0; i = i-2)
        Push (&b, keys [i]);

Printf (" First List:  ");
PrintList (a);
Printf (" second List : ");
PrintList (b);
struct Node* head = shuffle Merge (a, b);
Printf (" After Merge: ");
Print List (head);
return 0;
}
```

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 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 [+ + top] = x;
}
int s2 empty ()
```

```
{
      if (top 2 = = -1)
        return 1;
      else
        return 0;
}
int   s2 top ()
 {
      return s2 [top];
 }
      int s2 pop ()
  {
       Top-- ;
  }
  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 ; i<n; i++)
{
  scanf ("%d", &e);
  s1 push (e);
}
Printf ("enter the value of Constant sum:\
Scanf ("%d" &k);
 Printf (" The combination whose sum is eq
                              to k is: \n");
  sum (k);
}
```

(4) Write a Program to Print the elemts in a queue (i) in reverse order (ii) alternate order

(i)

```
# include <stdio.h>
# include "stack.h"
# include " QQ.h"
int main ()
{
```

```
int n , arr [20] i, j = 0;
~~stack stack s;~~
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(s));
      j++;
 }
 Printf (" Reverse");
 While (stop != -1)
```

```
            {
            Printf ("%d", Pop(&S));
            }
            Printf ("\n");
    return 0;
}
```

(ii)

```
  # include <stdio.h>
 # include <stdlib.h>
  struct node {
        int data;
        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
                                (size of(struct Node));
     new-node -> data = new-data;
  }
  int main()
  {
     struct node * head = NULL;
     Push ( & head, 12);
     Push(& head, 29);
     Push ( & head , 11);
     Push (& head, 23);
     Push(& head, 8);
     Print node (head) ;
```

)

(i) How array is different from the linked list

(ii) ~~which~~ ~~a~~ 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 toget {4, 1, 2, 3} as output for list 1 and {5, 6} for list 2.

i) 1) An Array is a data structure that contains a collection of similar type data elements where as the linked list is considered as non Primitive data structure contains a collection of Unordered linked elements known as nodes.

2) In the array the elements belong to indexes i.e if you want to get into the fourth element you have to write the variable name with its index or location within the square bracket

(3) In a linked list through, you have to start from the head and work your way through until you get to the fourth element.

~~(4) Accord~~

(4) Accessing anelement in an array is fast, while in linked list takes linear time, so ~~[illegible]~~. it is quite a bit slower.

(5) operations consume like insertion and deleti in array consume a lot of time. On the other hand the performance of these operations.

(6) In a array memory is assigned during compile time while in linked list it is allocated durning execution or runtime.

(5)
(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 * newnode = (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 hode* temp = head;
        while (temp = NULL)
        {
            Printf ("%d", temp -> data);
            temp P = temp -> next;
        }
} Printf (" \n")
```