N.Aasrith Varma.
AP19110010528
CSE-H

Q1) WAP to Insert and delete an element at the nth and kth Pointer in a limited list where n and k are given by user

Ans

```
#include <stdio.h>
# include <stdlib.h>
 Struct node {
   int data;
   Struct Node * next;
};

Struct Node * head;
 Void Insert (int data , int n ) {
Node * temp = newNode ();
 temp → data = data;
temp → next = Null;
if (n == 1) {
  temp → next = head;
  head = temp;
  return;
}
Void delete (int k ) {
 struct Node * temp = head;
 if (k == 1) {
 head = temp → next;
free (temp);
 return;
}
 Node * temp = head;
  for(int i=0 , i< n-2   , i++) {
     temp = temp -> next
  }
   temp -> next = temp -> next
   temp -> next = temp;
}
 Void Print ( );
for ( int i=0, i< K-2 , i++).
temp = temp -> next
free(temp );
}
int main () {
int n, x, k
head=Null
Printf ("Enter the position for inserting ");
 scanf ("%d" &n );
scanf("%d" ;& x);
Insert (x,n);
Printf ("Enter the position to delete);
Scanf ("%d", &k)
 Delete (k);
```

```
Printf (x)
 return;
 }
```

Q2) Construct a new linked list by merging alternate nodes of two lists.

Ans)

```
#include <stdio.h>
#include <stdlib.h>
Struct Node
{
  int data;
  struct Node* next;
};

Void Printlist (struct Node * head)
{
 struct Node * Ptr = head;
 While (Ptr)
 {
   Printf ("%d -> ", Ptr -> data);
   Ptr = Ptr -> next;

}
 Print f("Null \n");

}
```

```
Void Push (Struct Node ** head, int data)
{
 Struct Node* newNode = (struct Node*) mallou(size of (struct Node));
 newNode -> data = data ;
 New Node -> next = * head ;
 * head = newNode;
}

struct Node * ShuffleMerge (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 [ ] = { 11, 12, 13 , 14, 18, 20, 21 };
  int n = size of (keys) / size of (Keys [0]);
 struct Node *a = Null, *b= Null;
 for (int = n-1 , i>=0; i= i-2)
    Push (&a, Keys[i]);

 for (int i = n-2; i>=0; i= l-2)
    Push (&b; Keys [i]));
```

```
Print f(" First list  : ");
Print list (a);
Printf ("Second List  : ");
Printlist (b);

Struct Node* head = shuffle Merge (a, b);
Print f("After Merge : ");
Print list (head);

return 0;

}
```

Q3) Find all the elements int the stack whose sum is equal to K.

```
# includ<stadio.h>
int top=-1.
int x;
Char stack [100];
Void Push (int x);
Char Pop ();
int main ()
{
int i, c, b, a, k, f, sum=0, Count = 1;
Print f("Enter the number of elements ");
```

```
scanf ("%d", &c)
for (i=0 ; i<c ; i++) {
Printf ("Enter next element");
Scanf ("%d", &b);
Push (b);
}
Printf ("Enter the sum to be Checked");
Scanf ("%d", & K);
for (i = 0; i<c ; i++)
{
a = Pop ();
sum +=a
Count += 1;
if(sum == K) {
for (int J=0 ; J<Count, J++)
Printf ("%d", stack[J]);
f=1;
break;
}
Push (a);
}
if (f!=1)
```

```
Printf ("The elements in the stack dont add up to the sum");
}
Void Push(int x)
{
 if(top==99)
{
 Printf("\n stack is full \n");
return;
}
top=top+1
Stack[top==-1)
{
 Printf("\n stack is empty \n");
 return0;
}
 x= stack[top]
top = top-1
  return x;
}
```

Q 4) WAP to print the elements in a queue

i) in reverse order

(ii): in alternate order

Ans)

```
# include <stdio.h>
# define Size 10
 Void insert (int);
 Void detete ();
 int queue [20], f=-1, r=-1;
 Void main () {
     int Value, Choice;
     while (1) {
        Printf ("\n\n *** Menu *** \n");
        Printf ("1. Insertion \n 2. Deletion \n 3. Print Reverse \n
                                          4. Print alternate \n 5. Exit)
      Printf ("\n Enter your Choice: ");
       Scanf ("%d", &Choice);
 Case 1: Printf ("Enter the Value to insert: ");
 Scanf ("%d", &Value)
  insert (Value);
 break;
```

~~Case~~

```
Case 2: delete ()
break;
Case 3;
            Printf ("The Reversed queue is: ");
            for(int i= size ; i>=0 ; i--)
{
            if (queue[i]= =0)
            continue ;
            Printf ("%d", queue[i]);
 }
            break;

      Case 4;
         Printf ("Alternate elements of the queue are:");
         for (int i=0 ; i<size ; i+=2)
 {
         if (queue[i]==0)
         continue;
         Printf ("%d" queue [i]);
 }
        break;

Case5: exit(0);
 default : Printf ("\n Wrong selection ");
        }
```

```
}
}
Void insert (intValue ){
  if ((f==0 && r== size -1) || f==r+1)
  Printf ("\n Queue is full ");
  else{
    if (f==-1)
f= 0;
  r=(r+1) % size;
  queue [r]= Value;
  Printf ("\n Insertion successful");
}}

Void delete (){
   if (f==-1)
     Printf ("\n Queue is empty ");
 else{
   Printf ("\n Deleted : %d ", queue [f]);
   f = (f+1) % size;
   if (f==r).
 f= r= -1;

 }
}
```

Q5) (i) How array is different from linked list.
(ii) W.A.P to add the first element of one list to another list.

Ans)

(i) The 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 reference where each node consists of the data and the reference to the previous and next element.

(ii)

```
#include <stdio.h>
#include <stdlib.h>

struct Node
{
  int data;
  struct Node * next;
};

Void Printlist (struct Node * head)
{

    struct Node * Ptr = head;
    While (Ptr)
```

```
{
    Print f("%d -> ", Ptr -> data);
    Ptr = Ptr -> next;
}
  Print f("Null |n");
}
Void Push (Struct Node** head, int data)
{
   struct Node * new Node = (struct Node *) mallou(size of (struct Node));
   new Node -> data = data
   new Node -> next =* head;
   * head = new Node
Void Movenode (Struct Node ** dest Ref, struct Node ** Source Ref)
{
  if (* Source Ref == Null)
      return;
  struct Node * new Node = * Source Ref;
  * Source Ref = (* Source Ref) -> nexct;
  new Node -> next = * dest Ref;
   * dest Ref = new Node;
}
int main(Void)
{
  int Keys[]= {4,5,6};
  int n = size of (keys) / size of (keys [0]);
  struct Node* b= Null;
  for(int i=0; i<n; i++)
     Push(&b, 2* keys [i]);
 move Node (&a, &b);
 Print f ("first List : ");
 Print list(a);
 Printf ("Second list : ");
 Print list (b);
 return 0;
}
```