M. Chandrahas
AP19110010636 (CSE-F)

1)
```
# include <stdio.h>
# include <stdlib.h>
Struct node {
Struct Node * next;
};
Struct node * head;
void Insert( int data, int n){
Node * temp = new Node();
   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");
scanf ("%d", &n);
Scan f ("%d", &x);
Insert (x, n);
Printf(" enter the pos to delete");
[illegible]
Scan f("%d", &k);
Delete (k);
Print (x)
return ;
}
```

2)
```
# include <stdio.h>
# include <stdlib.h>
Struct Node {
    int data;
    Struct Node next;
}
void Print_list (Struct node * head)
{
   Printf("%d", (ptr_ data));
   ptr = ptr -> next;
   Printf ("Null\n");
}
void Push (Struct node* head, int data)
{
   Struct node* new = Struct node * malloc
                               (Size of struct);
   new -> data = data;
   new -> next = * head;
        * head = new;
}
Struct node * merge (Struct node *a,
                          Struct node * b)
{
    Struct Node temp;
     Struct Node * tail = & temp;
     temp . next = null;
     while (1) {
          if (a = = Null)
          {
             tail -> next = b;
             break;
          }
          else if (b = Null)
          {
              tain -> next = a;
              break;
          }
          else :
          {
              tail -> next = a;
              tain = a;
               a = a -> next;
               tail -> next = a;
          }
     }
      return fake next;
}
Void main()
{
    int keys [] = {1, 2, 3, 4, 5, 6, 7}
    int n = Size of (keys) / Size of keys[0]
3) # include < stdio.h>
 int top= -1;
 int x;
 char stack [100];
 void Push(int x);
 char PoP();
 int main()
 {
    int i, n, a, t, k, f, Sum=0, count=1;
    PrintF("enter no. of elements");
    ScanF("%d, & n);
    for(i=0; i<n; i++){
        PrintF("enter next element")
        ScanF("%d", & a);
        Push(a);
    }
    PrintF("enter the sum to check");
    ScanF("%d", & t);
    for (i=0; i<n; i++){
       t = PoP ();
       Sum += t;
       count += 1;
       iF(Sum == k){
       For (int j=0; j<cnt; j++)
       Print+("%d", Stack[j]);
       f = 1
       break;
     }
     Push
    }
   iF(1=1)
   Print("elements in stack dont add
                              to sum");
  }
 void Push (int x)
 {
 iF( top == 99)
 {
  printF("\n Stack is full! \n")
  return;
 }
 top = top +1;
  stack[top]= x;
 }
 charPoP()
 {
 iF(Stack (Top) == -1)
 x = Stack [top];
 top = top -1;
 return x;
}
```

```
1)
  struct node *a = Null, *b = Null;
  for(i=n-2; i>=0; i=i-2)
  Push (& b, Key[i]);
  struct Node* head = merge(a, b);
   Print_list(head);

 }
```

```
4) # include <stdio.h>
  # define size 10
  void insert (int);
  void delete();
  int queue[10], f = -1, r = -1;
  voidmain ()
  {
  int value, choice;
  while()
  {
  Print f("\n menu\n");
  Print f(" 1. Insertion \n 2. Deletion \n. 3
            3. Reverse  \n 4. Alternate")
  Print F("\n Enter choice.");
  ScanF("%d", & choice);
  Switch(choice)
  {
    case1: Print+( "enter the value");
           ScanF ("%d", & value);
           insert (value);
           break;
    case2: delete();
           break;
    case3: PrintF("Reverse of value:");
            for(i = Size; i>=0 ; i--)
            {
              if (queue[i] = =0
              continue
              Print ("%d" queue[i]);
            }
            break;
    case4: PrintF("another element");
             for(i=0; i<size; i+=2)
             {
               if (queue[i] = -0)
               continue;
               Print+("%d' queue(i));
             }
             break;
    case5: exit (0);
    default: Print ("wrong Selection"); }
```

Scanned with CamScanner

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* new_node = (struct node*)
      malloc ( size of struct)
  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 )
  {
    Print f ( "%d", temp -> data);
    temp = temp -> next;
  }
  Print( "\n" )
```

4) }}
```
void insert ( int value) {
 if (if = = 0 && r = = size -1) ||
                              f = = r+1)
 Print f( "\n Queue is full");
 else {
   if (f = = -1)
   f = 0;
   r = (r + 1) % size;
   Queue [r] = value;
   Print f("\n Insertion Success");
}}
void delete( )
{
 if (f = = -1)
 Print f( "\n Queue is empty");
 else
 {
   Print f( "\n Delete: %d", queue[f]);
   F = (f+1) % size;
   F = r = -1
 }
}
```

i) The major difference between Array and linked list is about their structure. Arrays is one index based data structure where each element associated with an index. on the other hand. wherea list relies on reference to the previous and next element.