Assignment-4

G. Karthik
AP19110010829
CSE-F

1) Write a Program to insert and delete an element at the nth position in a linked list where n or k is taken from user

A)
```
#include <stdio.h>
#include <stdlib.h>
Struct node {
 Int data;
 struct node* next;
};
Struct node* head;
Void insert (int data, int n) {
 node* temp = new code;
 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;
trace(temp);
}
int main() {
int n, x, t
head = Null;
Printf("Enter the position for inserting");
Scanf("%d", &n);
Scanf("%d", &x);
Insert(x, n);
Printf("Enter the Position todelete);
Scanf("%d", &k)
delete(k);
Print(x);
return;
}
```

2) Construct a new linked list by merging alternate nodes of two list for example In list 1 {1,2,3} and in list 2 {4,5,6} in new list we should have {1,4,2,5,3,6}

A)
```
#include <stdio.h>
#include <stdio.h>
    struct node{
     int data;
     Struct node* next;
     }
     void Print list (struct node* head)
     {
      Print f ("%d->", (ptr -> data));
       Ptr = Ptr -> next)
       Printf ("null\n");
      }
      void Push (struct node* head, int data)
     {
      Struct node* new = (struct node*) malloc
            (size at (struct node));
      new -> data = data;
      new -> next = *head;
       *head = new;
```

```
struct node* merge (struct node* a, struct node* b)
{
    struct node take;
    struct node * tail = take;
    take.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;
    }
}
    return take.next;
}
    void main ()
    {
int keys [] = {1, 2, 3, 4, 5, 6, 7}
```

```
int n= size of (keys) /size of keys[0]
  struct node * a=null ; * b= null )
  for (int i=n-1; i>0 ; i=i-a)
      Push (& a, keys [i ]);
  for (int i=n-2; i>=0 ; i=i-2)
      Push (& b; keys [i ]);
  Struct node *head =merge (a, b);
  Print list (head)
}
```

3) Find all the element 5 in the stack whose sum is equal to k

A) # include <stdio.h>

```
Void find (int arr[ ], int a, int K) {
  int total =0
  int x=0, y=0;
  for (x=0; x<a, x++) {
  while (total <K, & & y<a)
      total= arr [y]
      y++ ;
      If (total = =0 )
  {
    Printf (" find ");
  return;
}
```

```
total = arr[x];
}
}

int main(void){
    int arr[] = {9,10,12,4,1,2};
    int k=565;
    int a = sizeof(arr)/sizeof(arr[0]);
    find(arr,a,k);
    return 0;
}
```

---

4)(i) Implement queue in reverse order

```
#include<conio.h>
#include<stdio.h>
#define max 20
Void show(int stack[], int size, int top)
{
int i;
for(i=0; i<size; i++)
{
Printf("\n value at %d is %d", top, stack[top]);
    top=top-1;
}
}
void reverse(int stack[], int queue[], int *r,
                                    int *y,
                                    int *f)
```

```
{
  *f = 0;
  while (*r > -1)
  {
    *r = *r + 1;
    queue[*r] = stack[*f];
    *f = *f + 1;
  }
  while (*f <= *r)
  {
    *t = *t + 1;
    stack[*t] = queue[*f];
    *f = *f + 1;
  }
}
int main()
{
  int size
  int ibiem, t, i, stack[max], queue[max
  int top = -1, front = -1, rear = -1;
  Printf("Enter size of stack");
  Scanf("%d", &size);
  for(i=0; i<size; i++)
  {
    top = top + 1;
    Printf("Enter value position %d", top);
    Scanf("%d", &item);
```

```
stack[top] = item;
}
show(stack, size, top);
reverse(stack, queue, &top, &rear, &front);
    printf("\nAfter reverse:--");
        show(stack, size, top);
        getch();
    }
```

(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(structnode **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);

  return 0;
  }
```

5) (i) How array is different from link list

Array	linked list
An array is collection of element of similar data type	linkedlist is ddered collection of element of same type in each element connect using Pointers
2) Array Element can be accesed randomly in array index	2) Random accessing is not possible in link list element will accesed sequently
3. Data element are stored in contiguous location in memory	3. New element can be sorted anywhere a reference is created For new element using Pointers

(ii)

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
};void Push {struct node *head-ref,
int new-data)
{
structnode *new-node = (structnode*) 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");
}
```