Assignment - 4

Nitin Varma Rudraraju
AP19110010336
CSE-F

1) Write a program to insert and delete an elements at the nth and kth position in a linked list where n and 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 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;
   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 to delete");
  scanf ("%d", &k)
  delete (k);
  print (x);
  return;
}
```

12) Construct a new linked list by merging alternate nodes of two lists 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 <stdlib.h>
  struct node {
int data ;
struct node * next ;
}
void print list (struct node * head)
}
 print f ("%d→", (ptr→ data));
   ptr = ptr → next ;
   print f ("null/n");
}
void push (struct node * head , int data)
{
 struct node * new = (struct node *) malloc
          (size of (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 = tak..
```

```
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 key [] = {1, 4, 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 elements 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[] = [9,10,12,4,1,2,3]
  int k = 65;
  int a = size of(arr)/size of(arr[0]));
  find(arr,a,k);
  return 0;
}
```

i, implement queue in reverse order

```
#include <stdio.h>
#include <stdio.h>
#include <conio.h>
#define max 20
  void show (int stack [], int size, int top)
{
  int i;
  for (i=0; i<size; i++)
  {
  printf("in value at '%d is '%d, top, stack(top);
       top = top-1;
  }
  }
  void reverse (int stack[], int queue [], int *f, int *r, int *t)
{
  *f = 0;
  while (*t > -1)
    {r = *r +1
    queue [*r] = stack [*t];
      *t = *t-a;
    }
  while (*f <= *r)
  {
    *t = *t++1;
  stack [*t] = queue [*f];
    *f = *f+1;
  }
  }
  int main
```

```
{
int size
int item, 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 ("\n After reverse \n");
      show (stack, size, top);
      getch ();
   }
```

ii)

```
# include <stdio.h>
# include <stdlib.h>
    struct node {
    int data;
    struct node * next;
   3
   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);
{
struct node *new_node = (struct node)
    malloc (size of (strat 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, 120);
push (&head, 11);
push (&head, 23);
push (&head, 8);
return 0;
}
```

5 (i) How array is different from list

Array	linked list
- An Array is collection of element of similar data type	- linked list is odered corection of element of same type of in each element correct using pointers
- Array element can be accesed randomly in array index	- Random accesing is not possible in link list element will accesed sequently
- Data element are stored in contiguous location in memory	- New element can be sorted any where 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);
{
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");
}
```