Sk. Kishwar Parveen
AP19110010545
CSE-G

① Write a program to insert and delete an element at the n^{th} and k^{th} position in a linked list where n and k is taken from user.

Sol:-

```
# include <stdio.h>
# include <stdlib.h>
 struct node
 {
 struct node * next;
 };
 struct node *curr, * temp;
 void input (struct node*)
 Void delete (struct node *)
 void main (void)
 {
 struct node *s;
 int n;
 S= Null;
 do
 {
  Print f (" Enter The element to
                    insert ; \n;");
```

```
Print f ("2. Delete\n");
Print f ("3. Exit \n");
Print f ("Enter the choice;");
        temp -> next = curr -> next;
        curr -> next = temp;
        break;
      }
    }
}
void delete (struct node * x)
{
   int pos, c = 1;
   curr = 2;
  Printf ("Enter the element to be delete;");
  Scanf ("%d", d pos);
 while (curr -> next != Null)
 {
 c++;
 if (c == pos)
 {
 temp = current -> next;
 curr -> next = curr -> next -> next;
 free (temp)
```

```
}
curr = curr->next;
Push(&P, 3);
Printf("first linked list:\n");
Print list (R);
Push (&q,4);
Push (&q,5);
Push (&q,6);
Print f("second linked list:\n")
Print list(q);
merge(P,&q);
Print f("modified first linked list=\n");
Print list(P);
Print f("modified second linked list=\n");
Print list (q);
return 0;
}.
```

Out Put:-

1. Insert
2. Delete
3. Display
4. Exit

Enter ch: 1

Enter inserting position: 1

Enter value: 5

Enter ch: 1

Enter inserting position: 2

Enter value: 10

Enter ch: 3

5

10.

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

Sol:-
```
#include <stdio.h>
#include <stdlib.h>
// Datastructure to ptr. a linked list.
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;
    }
    Printf ("NULL/n");
```

```
// Insert newnode in begining
void Push (struct Node** head, int data)
{
  struct Node* newNode = (struct Node*)
                malloc (sizeof(struct Node));
  newNode->data = data;
  newNode->next = *head;
  *head = newNode;
// Function to construct a linked list by
   merging alternate nodes.
// two given linked lists using dilbar node
struct Node* ShuffleMerge (struct Node* a,
                           struct Node* b)
{
  Struct Node dilbar;
  Struct Node* tail = &dilbar;
  dilbar.next = NULL;
  while(1)
  {
    //empty list cases.
    if (a == NULL).
```

```
{
  tail -> next = b;
  break;
}
}
// move two nodes to tail
else
{
  tail->next = a;
  tail = a;
  a = a->next;
  tail->next = b;
  tail = b;
  b = b->next;
}
}.
return dibar.next;
}
int main(void)
{
  int keys[] = {1,2,3,4,5,6,7};
  int n = sizeof(keys) / sizeof(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));
Print ("first List :");
Print list (a);
Print f("second list :");
Print list (b);
Struct Node* head = Stuffle Merge (a,b);
Print ("After Merge :");
Print list (head);
return 0;
}
```

Out put:-

First List : 1->3->5->7->null

Second List : 2->4->6->null

After Merge : 1->2->3->4->5->6->7->null

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 the number of elements
in the stack");
scanf ("%d",&n);
for(i=0; i<n; i++){
printf("Enter next element");
scanf ("%od",&a);
push(a);
}
printf ("Enter the sum to be checked");
scanf("%d",&k);
for(i=0; i<n; i++)
{
```

```
t=pop();
sum+=t;
count+=1;
if(sum==k){
for(int j=0; j<count; j++)
printf("%d", stack[j]);
f=1
break;
}
push(t);
}
if(f!=1)
printf("The elements in the stack dont
 add up to the sum");
void push(int x)
{
if(top==99)
{
printf("\nStack is FULL!!!\n");
return;
}
top=top+1;
stack[top]=x;
```

```
}
char pop()
{
if (stack[top]==-1)
{
printf("\nstack is EMPTY!!!\n");
return 0;
}
x=stack[top];
top=top-1;
 return x;
}.
```

Out Put:-

```
Enter next element 34
Enter the sum to be checked 46
The elements in the stack dont add up to
the sum sktop/DSA Assignment 4$:/open.
Enter x[next element]x the number of elements
in the stacks"
enter next element 4
enter next element 3
Enter next element 5
Enter next element 3
enter next element 2
Enter the sum to be checked 10
3 5 3 2 0 t3jas_lok3sh@T3JAS-Linux-PC:~/Desktop
```

(4) Implement of queue in Reverise Order

Sol:-

```
# include<stdio.h>
# define SIZE 10
void insert(int);
void delete();
int queue[10], f=-1, r=-1;
void main(){
  int value, choice;
  while(1){
    printf("\n\n***** MENU *****\n");
    printf("1.Insertion\n2.
  Deletion\n3.Print Reverse\n4.Print
  Alternate\n5.Exit");
    printf("\nEnter your choice:");
    scanf("%d",&choice);
    switch(choice){
case 1: printf("Enter the value to be
  insert:");
scanf("%d",& value);
insert(value);
 break;
 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]);
}
case 4;
    printf("Alternate elements of the
     queue are:");
    for(int i=0;i<SIZE;it=2).
{
    if (queue[i]==0)
    continue;
    printf("%d",queue[i]);
}
    break;
  case 5: exit(0);
 default: printf("\n Wrong selection!!!
                  Try again!!!");
```

```
    }
}}
void insert(int value){
 if((f==0&&r==SIZE-1)||f==r+1)
  printf("\nQueue is Full!!! Insertion is
 not possible!!!");
 else{
  if(f==-1)
f=0;
  r=(r+1)%SIZE;
   queue[r]=value;
   printf("\nInsertion success!!!");
}}
 void delete(){
 if(f==-1)
  printf("\nQueue is Empty!!!
  Deletion is not possible!!!");
 else{
   printf("\nDeleted : %d",queue[f]);
   f=(f+1)%SIZE;
  if(f==r)
 f=r=-1;
 }}.
```

Out Put :-

```
***** MENU *****
1. Insertion
2. Deletion
3. Print Reverse.
4. Print Alternate
5. Exit.
Enter your choice: 3
The Reversed queue is: 6 5 4 3 2 1

***** MENU *****
1. Insertion
2. Deletion
3. Print Reverse.
4. Print Alternate
5. Exit
Enter your choice: 4
Alternate elements of the queue are: 1 3 5
```

5 (i) How are arrays different from linked list?

Ans: The major difference between arrays and linked list, is the way that they are structured

Arrays are an index based linear data structure, under each element is assigement a unique index to identify it.

Linked list on the other hand on refere-nces present in each of the list which both the precinos and next elemention the list.

Array's are a set of similar data descots which we started all in a row in the memory

Linked list is a data structure which is present in different parts of the memory, let each element has the memory adding if into adfacent elements

(ii) Program to add first node of linked list to another linked list.

Sol:

```
# include <stdio.h>
# include <stdlib.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;
  }
 print ("NULL \n");
 }
void Push (struct Node * * head, int data)
{
```

```
Struct Node* newNode=(struct Node*)
        malloc(size of(struct Node));

 new Node ->data=data;
 new Node ->next=* head;
 *head = new Node;
}

// Function take the node from the front
of source.

// and move it to front of desfination

void move Node(Struct Node** des(Ref,
                Struct Node** Source Ref)
{
 if (*Source Ref==NULL)
     return;
  Struct Node* new Node =* Source Ref;
*
 Source Ref =(*Source Ref)->next;
 new Node ->next = * dest Ref;
*
  dest Ref = new Node;
}.
```

```
int main(void)
{
    int keys[] = {1, 2, 3};
    int n = sizeof(keys)/sizeof(keys[0]);
    struct Node* a = NULL;
    for (int i = n-1; i >= 0; i--)
        push(&a, keys[i]);

// construct 1st linked list

// construct 2nd linked list
struct Node* b = NULL;
for (int i=0; i<n; i++)
    push(&b, 2*keys[i]);

// move front node of b and move it to
   the front of a.
moveNode(&a, &b);
Printf("first list = ");
PrintList(a);
Print("second list = ");
Print list(b);
return 0;
}
```

Out Put :-

First List : 6 → 1 → 2 → 3 → null

Second List : 4 → 2 → null