G. Anuhya
AP19110010462
CSE-F

ASSIGNMENT-4

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?

Code :-

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

struct node {
    int value;
    struct node *next;
};

void insert();
void display();
void delete();
int count();

typedef struct node DATA_NODE;

DATA_NODE *head_node, *first_node, *temp_node=0,
*prev_node, *next_node;

int data;

int main() {
    int option = 0;
    printf ("Slighly linked list Example- All Operations\n");
```

```
While (option <5) {
    Printf ("\n Options \n");
    Printf ("1: Insert into Linked List \n");
    Printf ("2: Delete from linked list \n");
    Printf ("3: Display Linked List \n");
    Printf ("4: Count Linked List \n");
    Printf ("Others : Exit() \n");
    Printf ("Enter Your option:");
    scanf ("%d", &option);
    switch (option) {
    case 1:
        insert ();
        break;
    case 2:
        delete ();
        break;
    case 3:
        display ();
        break;
    case 4:
        count();
        break;
    default:
        break;
    }
}
return 0;
```

```
Void insert () {
   Printf ("\nEnter Element for insert Linked list :\n");
   Scanf ("%d", &data);

   temp_node =(DATA_NODE*)
  malloc (size of (DATA_NODE));

   temp_node->value = data;

    if(first_node == 0) {
      first_node = temp node ;
    }
    else {
     head_node -> next = temp_node;
     fflush (stdin);
     }

void delete() {
  int countable , pos, i = 0;
  Countvalue = count();
  temp node = first_node;
  Printf("\n Display Linked list :\n");
  Printf("\n Enter position for Delete Element :\n");
    Scanf(" %d", &Pos);
     if (Pos >0 && pos <= countvalue ) {
     if (POS == 1) {
```

```
temp_node = temp_node -> next;
first_node = temp_node;
printf("\n Deleted successfully \n\n");
}
else {
While (temp_node != 0) {
 if (i == (Pos-1)) {
  Prev_node -> next = temp_node->next;
 if (i == (Pos-1)) {
   prev_node -> next = temp_node->next;
 if (i == (countvalue-1))
 {
 head_node = Prev_node;
}
  Printf ("\n Deleted successfully \n\n");
   break;
  } else {
   i++;
  Prev_node = temp_node;
  temp_node = temp_node->
next;
  }
 }
}
} else
  Printf ("\n Invalid Position \n\n");
}
```

```
void display() {
 int count = 0;
 temp_node = first_node;
 Printf("\nDisplay Linked List: \n");
 While (temp_node != 0) {
  Printf ("# %d #", temp_node->next;
}
 Printf ("\n No of Items In Linked list : %d\n",count);
}

 int count() {
   int count=0;
  temp_node = first node;
  while (temp node != 0) {
   count++;
   temp_node = temp_node->next;
 }
   Printf ("\n No of Items In linked list :
              %d \n", count);

   return count;

}
```

Output :-

```
Singly Linked List Example - All operations

options

1 : Insert into linked list
2 : Delete from linked list
3 : Display linked list
4 : Count linked list
Others : Exit ()
Enter Your Option: 1
Enter Element for Insert Linked list : 6

 options

1: Insert into linked list
2: Delete from linked list
3. Display linked list
4: Count linked list

  Enter Your Option : 1
  Enter Element$ for Insert linked list : 4
 options
1: Insert into linked list
2: Delete from linked list
3: Display linked list
4: Count linked list

  Enter Your option : 2
  Enter Position for Delete Element : 2

     Delete Successfully
```

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

Code :-

```
# 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;
   }
   Printf("Null \n");
}
 Void push (Struct Node ** head, int data)
{  Struct Node * new Node = (Struct Node*) malloc
                                (size o
     new Node → data = data;
```

```
    newNode->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[] = {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]);
    Printf("first list:");
    Print list(a);
    Printf("second list:");
    Print list(b);
  Struct Node* head = Shuffle Merge(a,b);
   Printf("After Merge:");
   Print list(head);
    return 0;
}
```

Input & Output

first list : 1 -> 4 -> 6 -> 8 -> null

second list : 2 -> 3 -> 5 -> 9 -> null

After merge : 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 8 -> 9 -> null

3) find all the elements in the stack whose sum is equal to k (where k is given from user).

code :-

```
#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("%d", &a);
Push(a);
}
Printf("Enter the sum to be checked");
Scanf("%d", &k);
for(i=0; i<n; i++)
{
t=pop();
sum+=t;
count++;
if(sum==k){
```

```
for(int i=0; i<count; i++)
Printf ("%d", stack[i]);
f=1;
break;
}
Push(t);
}
if (f!=1)
Printf("The elements in the stack don't add up to the
        sum");
}
Void push(int x)
{
if (top==99)
{
 Printf("\n stack is full !!!\n");
return;
top=top+1;
stack[top] = x;
}
 char pop()
 {
 if (stack[top]==-1)
 {
  Printf ("\n stack is EMPTY !!!\n");
  return 0;
 }
```

```
    x = stack [top];
    top = top-1;
    return x;
}
```

Output:

Enter the number of elements in the stack 4

Enter next element 4

Enter next element 2

Enter next element 5

Enter next element 1

Enter the sum to be checked 15

The element in the stack dont add up to the sum

4. Write a program to print the element in a queue

i. in reverse order.

ii. in alternate order.

```
#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\n 2.Deletion\n 3. Print Reverse\n
          4. Print Alternate\n 5. Exit");
  Printf("\n Enter Your choice : ");
  Scanf("%d", &choice);
  Switch (choice){
case 1: printf("Enter the value tobe insert:");
      scanf("%d",&value);
      insert(vale);
      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]);
    }
      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;
case 5 : exit(0);
default : printf("wrong selection!!! Try again!!!");
  }
}}
```

```
Void insert (int value){
    if ((f=0 && r = SIZE-1) || f=r+1)
      Printf("\n Queue is full!! Insertion is not
                                    Possible!!!");
    else{
      if(f==-1)
    f=0;
      r=(r+1) % SIZE;
      queue[r]=value;
      Printf("\n Insertion Success!!!");
    }}
    Void delete(){
     if (f==-1)
      Printf("\n Queue is Empty!!! Deletion is not
                                    Possible!!!");
     else{
      Printf("\n Deleted : %d", queue[f]);
       f=(f+1) % SIZE;
       if (f==r)
     f=r=-1;
    }}
```

Output :-

*** MENU ***

1. Insertion
2. Deletion
3. print Reverse
4. print Alternate
5. Exit

Enter your choice : 1
Enter the value to insert : 2

*** MENU ***

1. Insertion
2. Deletion
3. print Reverse
4. print Alternate
5. Exit

Enter your choice : 1
Enter the value to insert : 5

*** MENU ***

1. Insertion
2. Deletion
3. print Reverse
4. print alternate
5. Exit

Enter Your choice : 1
Enter the value to insert : 4

1. Insertion
2. Deletion
3. print Reverse
4. Print alternate
5. Exit

Enter Your choice 1
Enter the value to insert : 7

*** MENU ***

1. Insertion
2. Deletion
3. print Reverse
4. print Alternate
5. Exit

Enter Your choice : 3
Reverse que is 7 4 5 2

*** MENU ***

1. Insertion
2. Deletion
3. print Reverse
4. print Alternate
5. Exit

Enter Your choice : 4

Alternate elements of the queue are : 2 4

5) (i) How array is different from the linked list.

Difference between Array and linked list :-

Array and linked list the major 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 lists relies on references where each node consists of the data and the references to the Previous and next element.

(ii) Write a program to add the first element of one list to another list.

Code :-

```
#include <stdio.h>
#include <stdlib.h>
//Data structure to store a linked list node
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");
}
void push(struct Node** head, int data)
{
struct Node* newNode = (struct Node*) malloc(size
    of(struct Node));
  newNode->data = data;
  newNode->next = *head;
  *head = newNode;
}
void MoveNode(struct Node** destRef, struct Node**
                                        sourceRef)
 {
 if(*sourceRef == NULL)
   return;
struct Node* newNode =
*sourceRef; // the front source node
*sourceRef = (*sourceRef)->next;
  newNode->next = *destRef;
  *destRef = newNode;
 }
```

```
int main(void)
{
 int keys[] = {1,2,3};
 int n = sizeof(keys)/sizeof(keys[0]);

 struct Node* b = NULL;
 for(int i=0; i<n; i++)
   Push(&b, 2*keys[i]);

   MoveNode(&a, &b);

   Printf("First list: ");
   Print list(a);
   Printf("Second list:");
   Print list(b);
   return 0;

}
```

Output:-

first list: 6 -> 1 -> 2 -> 3 -> Null

Second list: 4 -> 2 -> Null