DSA ASSIGNMENT-4

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

```
#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, *previous_node,
*next_node;
int data;
int main()
{
  int choice=0
  while (choice<5){
  printf("\n choices \n");
  printf(" 1: Insert \n");
  printf(" 2: Delete \n");
  printf("3: Display \n");
  printf("4: Count Linked list \n");
  printf(" others : Exit() \n");
```

```
printf(" Enter your choice ");
Scanf (" %d", &choice);
Switch (choice) {
  Case 1: insert ();
          break ;
  Case 2: delete ();
          break;
  Case 3: display ();
          break;
  Case 4: count ();
          break;
  default:
          break;
  }
}
return 0;
}
void insert () {
  printf (" In Enter 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;
  }
```

```
    temp_node -> next = 0;
    head_node = temp_node;
    fflush(stdin);
}
void delete() {
int countvalue, position, n=0;
countvalue = count();
temp_node = first_node;
printf("\n Display Linked List : \n");
printf(" Enter position to Delete : \n");
scanf("%d", &position);
if(position >> 0 && &pos <= countvalue){
{
    if (position == 1)
    {
        temp_node = temp_node -> next;
        first_node = temp_node;
        printf(" Element deleted \n");
    }
    else
    {
        while (temp_node != 0)
        {
            if (i == (position - 1)
            {
                prev_node -> next = temp_node -> next;
                if (n == (countvalue - 1));
                { head_node = previous_node;
                }
                printf(" Deleted ");
                break;
            }
```

```
            else {
            {
              i++;
              previous_node = temp_node
              temp_node = temp_node -> next;
            }
          }
        }
      }
      else
      {
        printf(" Invalid \n\n");
      }
}
void display()
{
    int count = 0;
    temp_node = first_node;
    printf("\n Display : \n");
    while (temp_node != 0)
    {
      printf ("%d", temp_node -> value);
      count ++;
      temp_node = temp_node -> next;
    }
    printf ("\n No of Items : %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 %d \n", count);
  return count;
}
```

construct a new linked list by merging alternatives nodes of two lists 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}

```
#include <stdio.h>
# include <stdlib.h>
# include < assert.h >
Struct node
{
    int data;
    Struct node * next;
};
void move node(Struct node **x; Struct node **y);
Struct node *sorted merge( struct node *a, struct node *b);
{
   Struct node dall;
   Struct node * tail = & dall;
   dall.next = Null;
   while(1)
   {
      if (a==Null)
      {
         tail → next=b;
         break;
      }
      else if ( b==NULL )
      {
         tail → next=a;
         break;
       }
      if (a→ data <=b→ data)
      {
         move node { + (tail)→next), &a);
       }
      else.
```

```
        {
            move node = (&(tail) -> next, &b);
        }
        tail = tail -> next;
    }
    return (dall.next);
}
void move node *(struct node **x, struct node **y)
{
    struct node * newnode = *y;
    assort (newnode != NULL);
    * if = new node -> next;
    new node -> next = *x;
    *x == newnode;
}
void push (struct node ** head_ref, int new_data)
{
    struct node * new_node = (struct node *) malloc (size of (structnode));
    new_node -> data = new_data;
    new_node -> next = (* head_ref);
    (* head_ref) = new_node;
}
void point list (struct node * node)
{
    while (node != NULL)
    {
        printf ("%d", node -> data);
        node = node -> next;
    }
}
```

```
int main()
{
    Struct node * res = null;
    Struct node * a = NULL;
    Struct node * b = NULL;
    push(&a,1);
    push(&a,2);
    push(&a,3);
    push(&a,4);
    push(&b,5);
    push(&b,6);
    res = sorted merge (a,b);
    printf("merge linked list is : \n");
    printf list(res);
    return 0;
}.
```

3. Find all the elements in the stack whose sum is equal to k.

```
# Include <stdio.h>
int top = -1;
int a;
Char stack[50];
void push(int a);
Char pop();
int main()
{
    int n, i, x, t, k, f, sum=0, count = 1;
    printf("Enter the number of elements");
    Scanf("%d", &i);
```

```
for (n=0; n<i; n++)
{
  printf("Enter next element");
  scanf("%d", &a);
  push(a);
}
printf("Enter the sum tobe checked");
scanf("%d", &K);
for (n=0; n<i; n++)
{
  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);
}
printf("The elements in the stack don't add up");
}
void push(int a)
{
  if (top == 99)
  {
    printf("\nstack is FULL !!!\n");
    return;
  }
  top = top+1;
  stack[top] = a;
}
```

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

4. Write a program to print the elements in a queue
   i) reverse order
   ii) 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*** MENU ***\n");
        printf("1. Insertion\n 2. Deletion \n 3. Reverse \n 4. Alternative
                                        \n 5. Exit ");
        printf("\n Enter 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]);
    }
    break;
    case 4;
    printf(" Alternate elements of the queue : ");
    for(int i=0 ; i<SIZE ; i+=2)
    {
        if (queue[i] ==0)
        continue;
        printf(" %d", queue[i]);
    }
    break;
    case 5: exit(0);
    default: printf(" In Wrong selection ");
  }
}
}
```

```
void insert (int value)
{
    if ((f==0 && r==SIZE-1) || f==r+1)
      printf ("\n Queue is full");
    else
    {
        if (f==-1)
        f=0;
        r= (r+1) % SIZE;
        queue [r] = value;
        printf ("\n Insertion done");
    }
}
void detete ()
{
    if (f==-1)
      printf ("\n Queue is Empty !");
    else
    {
      printf ("\n Deleted : %d", queue[f]);
      f= (f+1) % SIZE;
      if (f==r)
      f=r=-1;
    }
}
```

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

(ii) Write a program to add the first element of one list to another list to another list of example we have {1,2,3} in list 1 and {4,5,6} in list 2 we have to get {4,1,2,3} as output for lists an {5,6} for list 2.

i) → difference in their structure

→ arrays are index based

→ linked list relies on reference to the previous and next element.

(ii)
```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node * next;
};
void push (struct node ** hed_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");
}
```