DSA- Assignment-4

M.Haritha
AP19110010270

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

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
Int data,
struct Node *next;
};
Struct Node*head,
Void Insert (int data, int n) {
Node* t = new Node();
t→ data= data;
 t-> next = null;
if(n==1) {
t →next= head;
 head=t;
 return;
}
Node* temp = head;
for(int i=0, i<n-1 ; i++){
temp=temp→next;
}
t→ next= temp→next;
temp→ next=t;
}
```

```
voidPrint ( );
void Delete n (int K) {
struct Node* temp=head;
if(K==1) {
head = temp→next;
free(temp);
return;
}.
for (int i=0 ; i<K-2;i++).
t = t→ next;
struct Node* temp= t→next;
t→next= temp→next;
free(temp);
}.
int main() {
int n,x,K;
head=null;
printf("Enter position and data to
 be inserted");
scanf ("%d", &n);
scanf ("%d", &x);
Insert (x,n);
printf("Enter the position to be
   deleted:");
scanf("%d", &k);
Delete (k);
{Print(n); }.
```

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

```
#include <stdio.h>
# include <stdlib.h>
struct Node
{
 int data;
 struct Node* next;
};
void printf(struct Node* top)
{
 struct Node * ptr= top;
 while (ptr);
 {
 printf("%d->"; ptr→data);
 ptr=ptr→next;
 }.
printf("NULL\n");
}.
void push(struct Node** top, int data)
{
struct Node * New Node= (struct Node*)malloc(size of (struct Node));
New Node →data= data;
New Node → Next= *top;
*top= New Node;
}
struct Node * alternate merge(struct Node*a, struct Node*b)
{
```

```
struct Node temp;
Struct Node* bottom = &temp;
temp.next = NULL;
While (1)
{
 if (a==NULL)
{
~~bottom→~~ next = b;
bottom→
break;
}.
else if (b==NULL)
{ bottom→next = a ; break ;}.
else
{ bottom → next =a;
 bottom =a;
 a=a→next;
 bottom→next =b;
 b=b→next;
}}.
return temp next;}.
 int main( )
{  int ~~ta~~ list[ ] = { 1,2,3,4,5,6,7,8,9,10}.
   int n= size of(list) / sinze of(list[0]);
   Struct Node*a = NULL,*b=NULL;
 for (int i=n-2; i≥0, i=i-1)
 Push (&a, list [i]);
for (int i = n-2 ; i>=0, i=i-1)
```

```
push (&b, list[i])
Struct Node* top = alternate
                  merge(a,b);
printf("After merging the new
                  list");
Printlist(top);
return 0;}.
```

```
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 no.of elements in 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+=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
        don't 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 empty\n");
   return 0;
  }.
  x=stack[top];
  top=top-1;
    return x:
 }.
```

```
#include <stdio.h>
 #define SIZE20
 Void insert (int)';
  Void delete();
  int queue [20], 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 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;
        case4:
            printf("Alternate elements of 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("\n Wrong selection!");
        }.
  }}
  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("Queue is Empty!! Deletion is not possible");
```

```
else {
        printf ("\n Deleted: %d", queue [f]);
        f =(f+1) %size;
        if (f==r)
    f= r=-1;
    }}.
```

5. How are array's different from linked list?

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

Array's	linked list.
Arrays are an idex based linear data structure, where each element is asigned a unique index to Identify it.	linked list on the other hand rely on references present in each node of the list which ~~reject~~ refers to ~~both~~ the previous and next element in the list.
Array's are a set of similar data dejects which are stored all in a row (sequentially) in the memory.	linked list is a data structure which is present in different part of the memory, but each element has the memory address of adjacent elements.
iii) Size of an Aaray is fixed	size of a list is not found
Insertion and Deletion takes more time.	Insertion and deletion process takes less time.

```
# 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(size of (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");
  }.
  void merge(struct Node * P, struct node **q)
  { struct Node * P=curr = P, *q - curr = *q;
    struct Node * P_next, *q.next;
while (P_curr != NULL && q-curr != NULL)
P_next = P_curr->next;
q-next = q-curr->next;
q-curr -> next = q-curr;
 P_curr= P_next;
 q-curr = q-next;
}.
*q=q-curr;
}.
int main()
{
  Struct Node * P=NULL, *q =NULL;
  push(&P, 6);
  push (&P, 5);
  Push(&P, 10);
  printf("1st linked list \n");
   Printlist(P);
  Push (&q, 9);
  Push(&q, 11);
  Push (&q, 13);
  Push (&q, 15);
  Printf("2nd linked list \n");
  Printlist(q);
  merge(P, &q);
```

```
printf(" changed lict 1\n");
Print list(P);
Print f(''changed list 2\n");
Printlist(q);
```

output.

```
1st linked list.
10 5 6
2nd linked list
15 13 11 9
changed list 1
10 15 5 13 6 11
Changed list 2
9
```