DSA Assignment-4

N. Praneeth Babu
AP19110010383
CSE-H

1. Write a program to insert & delete an element of the nth & Kth pointer in the linked list where n & k are taken from the user.

Ans:

```
#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;
}
```

Scanned with CamScanner

```
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;
 free(temp);
}
int main() {
int n,x,k;
head = null;
printf("enter the position for inserting: ");
scanf("%d", &n);
scanf("%d", &k);
insert(x,n);
printf("enter the position to delete");
scanf("%d", &k);
Delete(k);
print(x);
return;
}
```

Scanned with CamScanner

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

Ans:-

```
#include <stdio.h>
#include <stdlib.h>
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(sizeof
                                  (struct Node));
  newNode→data = data;
  newNode→next = *head;
  *head = newNode;
}
```

Scanned with CamScanner

```
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;
}
```

Scanned with CamScanner

```
int main (Void.)
{
    int keys[] = {1,2,3,4,5,6,7};
    int n = size of(keys)/size of(keys[0]);
    struct Node *a = NULL, *b = NULL;
    for(int i= n-1; i>=0; i=i-2)
        push(&a, keys[i]);
    for(int i=n-2; i>=0; i=i-2)
        push(&b; keys[i]);
    printf("First List:");
    printList(a);
    printf("Second List:");
    printlist(b);
    struct Node* head= shuffle Merge(a,b);
    printf("After Merge:");
    printList(head);
    return 0;
}.
```

Scanned with CamScanner

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

Ans:

```
#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 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+=1;
```

Scanned with CamScanner

```
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 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");
```

Scanned with CamScanner

```
return 0;
}
x = stack[top];
top = top - 1;
return x;
}.
```

Scanned with CamScanner

1. Write a program to print the element of 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(){
printf("\n\n** MENU **\n");
printf("1. Insertion \n 2. Deletion \n 3. 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;
```

Scanned with CamScanner

```
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 ("Alefernate elements of queue:");
      for(i=0; i<SIZE ; i+=2)
      {
       if(queue[i]==0)
       continue;
      printf ("%d", queue[i]);
      }
       break;
      case5: exit(0);
      default: printf("In wrong selection");
       }
}}
```

Scanned with CamScanner

```
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;
 }}
```

Scanned with CamScanner

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

Ans:(i) The major difference between Array & Linked list regards to their structure. Arrys are index based data structure. where each element associated with an index. On the other hand, Linked list relies on references where each node consits of the data & the reference to the previous & next element. Accessing an element in array is fast, while linked list takes linear time.

(ii)
```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
int data;
struct Node* next;
}
void printList (structNode* head)
{
struct Node* ptr = head;
while (ptr.)
```

Scanned with CamScanner

```
{
 printf("%d -> ", ptr->data);
  ptr = ptr->next;
}
void push(struct Node** head, int data)
{
struct Node* New Node = (struct Node*) malloc
                    (size of (struct Node));
new Node->data = data;
new Node->next= *head;
 *head = new Node;
}
void MoveNode(struct Node** destRef, struct Node**
                                    source Ref)
{
 if (*sourceRef == NULL)
  return;
 struct Node* new Node = *source Ref;
 *source Ref = (*source Ref)->next;
 new Node->next = *destRef;
 *destRef = new Node;
}
```

Scanned with CamScanner

```
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, key[i]);
struct Node* b = NULL;
for(int i=0; i<n; i++)
push(&b, *2keys[i]);
MoveNode(&a,&b);
printf("First List: ");
printList(a);
printf("second List: ");
printList(b);
return 0;
}.
```

Scanned with CamScanner