ASSIGNMENT-4.

M. SIRI ANJANI REDDY
CSE-(F)
AP19110010439.

1. Write a program to insert and delete an element of the nth and kth position in a list where n and k is taken from the 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
 {
 printf("Enter the element to insert;\n");
  printf("2. Delete\n");
  printf("3. Exit\n");
```


Scanned with CamS

```
printf("Enter the choice:");
scanf("%d",&n);
switch(n)
{
  case 1: input(s)
          break;
  case 2: delete(s)
          break;
 } while(n!=3)
}
void input(struct node *z)
{
  int pos, c=1;
  curr=z;
 printf("Enter the element to be inserted :");
 scanf("%d", &pos);
  while (curr->next!=NULL)
  {
  c++;
  if (c==pos)
  {
   temp=(struct node *) malloc(sizeof(structnode));
   printf("Enter the numbers:");
   scanf("%d", &temp->n);
```

Scanned with CamS

```
        temp -> next = curr -> next;
            curr -> next = temp;
        break;
        }
    }
}
void delete (struct node *z)
{
    int pos, c=1;
    curr = z;
    printf("enter the element to be deleted:");
    scanf("%d", &pos);
    while (curr -> next != NULL)
    {
        c++;
        if (c==pos)
        {
        temp = current -> next;
        current -> next = curr -> next -> next;
        free (temp)
        }
        curr = curr -> next;
        }
        void merge( struct node *p , struct node *q )
        {
```


Scanned with CamS

```
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 = p_next;
    p_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, 1);
    push (*p, 2);
    push (*p, 3);
    printf ("First linked list: \n");
     print list (R)
    push (*q, 4);
     push (*q, 5);
     push (*q, 6);
    printf ("second linked list: \n");
     print list (q);
```

Scanned with CamS

```
    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;
}
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;
```


Scanned with CamS

```
merge (p, *q);
  printf ("modified first linked list = \n");
  print list (p);
  printf ("modified second linked list : \n");
  print list (q);
  return 0;
}
```

(2) 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>
struct Node
  {
    int data;
    struct Node * next;
  };
void printlist ( struct Node * head)
  {
    struct Node * ptr = head;
    while (ptr)
    {
      printf ("%d ⟶ ", ptr→data);
```

Scanned with CamS

```
        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:");
    printlist(a);
printf("second list:");
```


Scanned with CamS

```
printlist(b);
struct Node * head = ShuffleMerge(a,b);
printf("After Merge:");
printlist(head);
return 0;
}
```

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

```
#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);
}
```

Scanned with CamS

```
printf(" enter the sum to be checked");
scanf(" %d ", &t);
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 upto
          the sum");
 }
 void push (int x)
 {
  if (top==99)
  {
   printf ("\n stack is full!!!\n");
   return;
  }
   top= top+1;
```


Scanned with CamS

```
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;
}
```

(4) Write a program to print the elements in a queue

(i) in reverse order

(ii) in alternate order.

Sol: (i)

```
#include <stdio.h>
#include "stack.h"
#include "QQ.h"
int main()
{
int n, arr[20], i, j=0;
struct stack s;
int stack(*s);
printf("enter number");
scanf("%d", &n);
```

Scanned with CamS

```
for (i=0; i<n; i++)
{
  printf ("enter values:");
  scanf ("%d", &arr[i]);
}
for(i=0;i<n;i++)
{
  insert (arr[i]);
}
  while (j!=n)
  {
    push (*s, del());
    j++;
  }
  printf ("Reverse is");
  while (stop!=-1)
  {
    printf(" %d", pop(*s));
  }
  printf("\n");
return 0;
}
```

(ii)

```
# include <stdio.h>
# include <stdlib.h>
struct Node {
  int data;
```


Scanned with CamS

```
    struct Node *next;
};
void printnodes (struct Node * head)
{
    int count = 0;
    while (head != NULL) {
        if (count %2 == 0){
            printf ("%d", head->data);
        }
        count++;
        head = head->next;
    }
}
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;
}
int main()
{
    struct node * head = NULL;
    push(*head, 17);
```

Scanned with CamS

```
push(*head, 23);
push(*head, 6);
push(*head, 20);
push(*head, 15);
printnode(head);
return 0;
}
```

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

(2) write a program to add first element of one list to another list of example we have {1,2,3} in list 1 and {4,5,6} in list 2 ....

Sol: (1) The major difference between Array and linked lists regards to their structure, arrays are based on index data structure where each element associated with an index on other hand linked lists relies on reference to the previous and next element.

(2)

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
```


Scanned with CamS

```
struct node * next;

};
void push (struct node * * head_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");
};
```

Scanned with CamS