E. Aadarsh Reddy
AP19110010471
CSE-H

ASSIGNMENT-4

① write a programe to intsert and delete an element at the n^{th} and k^{th} position in a linked list where n and k is taken from user.

Sol)
```
# incldede <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
{
Print f(" Enter The element to insert; \n; ");
Printf (" 2. Delete \n");
Printf(" 3. Exit \n");
Printf (" Enter the choice:");
Scanf ("%d", &n);
    Switch (n)
    {
    case 1: input (s);
            break;
```

Scanned with CamScanner

```
        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 (size (struct node));
        Printf(" Enter the number: ");
        Scanf("%d", &temp->n);
         temp-> next = curr->next;
         curr-> next = temp;
         break;
    }
  }
}
```

Scanned with CamScanner

```
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 = curr->next;
curr->next = curr->next->next;
Free (temp)
}
curr = curr->next;
}
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 = p_next;
```

Scanned with CamScanner

```
    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(P);
  Push (&q,4);
  Push (&q,5);
  Push (&q,6);
  Printf("Second linke list ;\n");
  Print list(q);
  merge (P,&q);
  Printf("modified first linked list=\n");
  print list (P);
  Printf(" modified second linked list = \n");
  Print list(q);
 return 0;
}
```

Scanned with CamScanner

(2) Construct a new linked list by merging alternate 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}

Sol)
```
# 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 dummy;
struct node * tail = & dummy;
dummy.next = Null;
while (1)
{
 if (a==* = Null)
{
        * y = newnode -> next;
        new node ->next = * x;
          * x-> new node;
 }
Void push (struct node ** head-ref , int new-data)
```

Scanned with CamScanner

```
{
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 * node)
{
while (node != Null)
{
printf("%d", node -> data);
node = node -> next;
}
}
    tail -> next = b;
    break;
}
else if (b == Null)
{
  tail -> next = a;
  break;
}
if (a -> data <= b -> data)
{
move node (&(tail) -> next, &a);
}
```

Scanned with CamScanner

```
else
{
  move node(&(tail->next, &b);
}
tail = tail->next;
}
return (dummy.next);
}
void moveNode(struct node **x, struct node **y)
{
  struct node *newnode = *y;
  assert(newnode != NULL);
int main()
{
  struct node* res = NULL;
  struct node* a = NULL;
  struct node* b = NULL;
  Push(&a, 1);
  Push(&a, 2);
  Push(&a, 3);
  Push(&b, 4);
  Push(&b, 5);
  Push(&b, 6);
  res = SortedMerge(a, b);
```

Scanned with CamScanner

```
Printf ("merge linked list is:\n");
Printf (list (res);
return 0;
}
```

Scanned with CamScanner

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

Sol)
```
#include <stdio.h>
int S1[10], top1 = -1, S2[10], top2 = -1;
int S1empty()
{
  if (top1 == -1)
      return 1;
  else {
      return 0;
}
int S1top()
{ return S1[top1];
}
int S1pop()
{  top1--;
}
int S1push(int x)
{ S1[++top1] = x;
}
int S2empty()
{
     if (top2 == -1)
         return1;
     else
         return;
}
int S2top()
{
  return S2[top2];
```

Scanned with CamScanner

```
}
int S2 pop()
{
  top2--;
}
int S2 push (int x)
{
  S2[++ top2] = x;
}
int Sum (int k)
{
   int x;
   while (S1 empty() != 1)
   {
      x = S1 top();
        S1 pop();
        while (S1 empty() != 1)
        {
         Printf ("%d, %d) \n", x, S1 top());
        }
        S2 push (S1 top());
           S1 pop();

     }
     while (S2 empty() != 1)
     {
       S1 push (S2 top());
        S2 pop();
     }
   }
}
int main()
{
```

Scanned with CamScanner

```
int n,i,e,k;
Printf("enter the no. of elements of stack:\n");
Scanf("%d", &n);
for(i=0, i<n, i++)
{
  Scanf("%d", &e)
    S, push(e);
}
Printf("enter the value of constant sum:\n");
Scanf("%d", &k);
Printf("The combinations whose sum is equal to k is:\n");
Sum(k);
}
```

Scanned with CamScanner

4) Write a program to print the element 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;
 initstack (&S);
 Printf("Enter no");
 Scanf("%d", &n);
 for (i=0; i<n; i++)
 {
  PrintF("Enter value:");
  Scanf("%d", & arr[i]);
 }
 for (i=0, i<n, i++)
 {
 insert(arr [i]);
 }
 While (j!=n)
 {
  Push (&S, del());
  j++;
 }
 Print ("Reverse is");
 While (stop != -1)
 {
```

Scanned with CamScanner

```
    Printf ("%d", Pop (&S));
}
PrintF("\n");
return(0);
}
```

ii)
```
#include <stdio.h>
#include <stdlib.h>
struct node{
  int data;
  Struct Node* next;
}
Void print node (Structur Node* head)
    {
      int count=0;
      while (head! =NULL){
        if ( count %2==0){
          Printf("%d", head -> data);
          }
          count++;
        head= head-> next;
      }
    }
Voide 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);
```

Scanned with CamScanner

```
    (*head_ref) = new_node;
}
int main()
{
    struct node* head = NULL;

        Push(&head, 12);
        Push(&head, 29);
        Push(&head, 11);
        Push(&head, 23);
        Push(&head, 8);
        printList(head);

    return 0;
}
```

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 of example we have {1,2,3} in list1 and {4,5,6} in list2 see how to get {4,1,2,3} as output for list 1 and {5,6} for list2

Sol) (i) The major different b/w Array and linked lists regards to their structure, arrays are index based data structure where each element associated with an index. On the other hand, 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 * * 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;
```

Scanned with CamScanner

```
}
Void print list (struct node * head)
{
struct node * temp = head;
while (temp != Null)
{
printf("%d", temp -> data);
  temp = temp -> next;
}
printf("\n");
}
```

Scanned with CamScanner