D. Tarun Venkat Sai
AP19110010504
CSE-H

(1) Write a program to insert and delete an element of the n^{th} and K^{th} position in a linked list where n and K is taken from 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
{
 Print f(" Enter the element to insert; \n;");
 Printf ("2. Delete \n");
 Printf ("3. Exit \n");
 Print f (" 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(size of (struct node));
    Printf (" Enter the numbers:");
    Scanf (" %d" ,&temp ->n);
          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 delete:");
  Scanf ("%d", &pos);
  while (curr -> next != Null)
 {
  c++;
  if (c == pos)
 {
  temp = current -> 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;
 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 linked list: \n");
  print list(Q);
  merge(P, &Q);
  Printf(" modified first linked list = \n");
  Printf List(P);
  Printf(" modified second linked list = \n");
  Print list(Q);
 return 0;
}
```

(12) Construct a new linked list by merging alternative 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}

A)

```
#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 = new node -> next;
     new node -> next = *x;
     * x = new node;
  }
  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 PointList(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)
  {
  moveNode(&(tail->next),a);
  }
  else
  {
    moveNode(&(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 = sorted merge(a,b);
  printf("merge linked list is: \n");
  print list(res);
  return 0;
}
```

(3) ~~Find all the elements in the stack~~

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

A)

```
#include <stdio.h>
int a1[10], top1=-1, a2[10], top2=-1;
int a1_empty()
{
  if(top1==-1)
    return 1;
  else
    return 0;
}
int a1_top()
{
  return a1[top1];
}
int a1_pop()
{
  top1--;
}
int a1_push(int n)
{
  a1[++top1]=n;
}
int a2_empty()
{
  if(top2==-1)
    return 1;
  else
    return 0;
}
int a2_top()
{
  return a2[top2];
}
int a2_pop()
{
```

```
    top2--;
}
int a2_push(int x)
{
    a2[++top2]=x;
}
int sum(int k)
{
    int x;
    while(a1_empty()!=1)
    {
        x=a1_top();
        a1_pop();
        while(a1_empty()!=1)
        {
            if(x+a1_top()=k)
            {
                printf("%d,%d)\n",x,a1_top();
            }
            a2_push(a,stop());
            a1_pop();
        }
        while(a2_empty()!=1)
        {
            a1_push(a2_top());
            a2_pop();
        }
    }
}
int main()
{
    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);
  a, 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);
}
```

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

i) in reverse order

ii) in alternate order

A) (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 no");
Scanf("%d",&n);
for(i=0; i<n; i++)
{
 print("Enter values:");
 Scanf("%d",&arr[i]);
}
 for(i=0, i<n, i++)
{
 insert(arr[i]);
}
```

```
while (j!=n)
{
push(&S, del[j]);
j++;
}
print ("Reverse is:");
while (stop! = -1)
{
Printf ("%d", pop(&S));
}
printf("\n");
return 0;
}
```

(ii)

```
# include <stdio.h>
# include <stdlib.h>
Struct node {
int data;
Struct node * next;
}
Void print nodes (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 (struct node));
```

```
new_node -> data = new_data;
new_node -> next = (*head_ref);
(*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);
  Print node (head);
  return 0;
}
```

(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 list 1 and {4,5,6} in list 2 we have to get {4,1,2,3} as output for list1 and {5,6} for list 2.

Sol: (i) The major difference b/w Array and linked lists regards to their structure, Arrays are index ~~reg~~ 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;
}
 void Print list (Struct node * head)
{
  Struct node * temp =head;
  while(temp!= Null)
 {
  Printf(" %d" , temp ->data);
   temp= temp->next;
 }
  Printf("\n");
}
```