WAP to Implement Single Link List to simulate Stack & Queue Operations.-

```
# include<stdio.h>
# include <stdlib.h>

struct node {
   int data;
   struct node *next;

};

struct node *top=NULL;
struct node *push();
struct node *pop();
struct node *display();

struct node *push()
{
   struct node *newnode;
   newnode=(struct node*) malloc(sizeof(struct node));
   printf("enter data : ");
   scanf("%d",&newnode->data);
   if(newnode == NULL)
   {
      printf("Memory not allocatede");
   }
   else if (top==NULL)
   {
      newnode->next=NULL;
```

```
        top=newnode;
    }
    else
    {
        newnode->next=top;
        top=newnode;
    }
}


struct node *pop()
{
    struct node *temp;
    temp=top;
    if (top==NULL)
    {
        printf("empty");
    }
    else
    {
        top=temp->next;
        free(temp);
    }

}

struct node *display()
{
    struct node *temp;
    temp=top;
    if (top==NULL)
```

```
    {
        printf("empty");
    }
    else
    {
        while (temp!=NULL)
        {
            printf("\t%d\n",temp->data);
            temp=temp->next;
        }

    }

}

int main()
{
    int option;
    while (1)
    {
        printf("------------Choose the operation --------------------\n");
        printf("\n1.push.\n2.pop.\n3.display\n4.Exit.");
        printf("\nEnter your option.");
        scanf("%d",&option);
        switch (option)
        {
        case 1:push(top);
            printf("\n CREATED");
            break;

        case 2:pop(top);
```

```
            break;

        case 3:display(top);
            break;

        case 4:exit(0);

        default:printf("wrong value");
            break;
        }
    }
}
```

Output:

```
----------Choose the operation ----------

1.push.
2.pop.
3.display
4.Exit.
Enter your option.1
enter data : 10

 CREATED----------Choose the operation ----------

1.push.
2.pop.
3.display
4.Exit.
Enter your option.1
enter data : 20
```

CREATED----------Choose the operation ----------

1.push.

2.pop.

3.display

4.Exit.

Enter your option.1

enter data : 30

CREATED----------Choose the operation ----------

1.push.

2.pop.

3.display

4.Exit.

Enter your option.1

enter data : 40

CREATED----------Choose the operation ----------

1.push.

2.pop.

3.display

4.Exit.

Enter your option.3

40

30

20

10

----------Choose the operation ----------

1.push.

2.pop.

3.display

4.Exit.

Enter your option.2

----------Choose the operation ----------

1.push.

2.pop.

3.display

4.Exit.

Enter your option.3

30

20

10

----------Choose the operation ----------

1.push.

2.pop.

3.display

4.Exit.

Enter your option.4

Queue

```
#include <stdio.h>
#include<stdlib.h>

struct node {
   int data;
   struct node *next;

};

struct node *f=NULL;
struct node *r=NULL;
struct node *enque();
struct node *deque();
struct node *display();

struct node *enque()
{
   struct node *newnode;
   newnode=(struct node *) malloc(sizeof(struct node));
   printf("enter data: ");
   scanf("%d",&newnode->data);
   newnode->next=NULL;
   if (f==NULL&&r==NULL)
   {
      f=r=newnode;

   }
   else
   {
      r->next=newnode;
```

```
      r=newnode;
   }
}

struct node *deque()
{
   struct node *temp;
   temp=f;
   if (f==NULL)
   {
      printf("empty");

   }
   else
   {
      printf("deleted element is: %d",temp->data);
      f=temp->next;
      free(temp);
   }
}

struct node *display()
{
   struct node *temp;
   temp=f;
   if (f==NULL)
   {
      printf("empty");

   }
   else
```

```
    {
        while (temp!=NULL)
        {
            printf("\t%d\n",temp->data);
            temp=temp->next;
        }
    }
}

int main()
{
    int option;
    while (1)
    {
        printf("----------------------Choose the operation------------------\n");
        printf("\n1.enque.\n2.deque.\n3.display\n4.Exit.");
        printf("\nEnter your option.");
        scanf("%d",&option);
        switch (option)
        {
        case 1:enque();
            break;

        case 2:deque();
            break;

        case 3:display();
            break;

        case 4:exit(0);
```

```
      default:printf("wrong value");
         break;
      }
   }
}
```

Output:

```
----------Choose the operation----------

1.enque.
2.deque.
3.display
4.Exit.
Enter your option.1
enter data: 10
----------Choose the operation----------

1.enque.
2.deque.
3.display
4.Exit.
Enter your option.1
enter data: 20
----------Choose the operation----------

1.enque.
2.deque.
3.display
4.Exit.
Enter your option.3
      10
```

20

----------Choose the operation----------

1.enque.

2.deque.

3.display

4.Exit.

Enter your option.2

deleted element is: 10----------Choose the operation----------

1.enque.

2.deque.

3.display

4.Exit.

Enter your option.3

20

----------Choose the operation----------

1.enque.

2.deque.

3.display

4.Exit.

Enter your option.4

Leet code challenge-

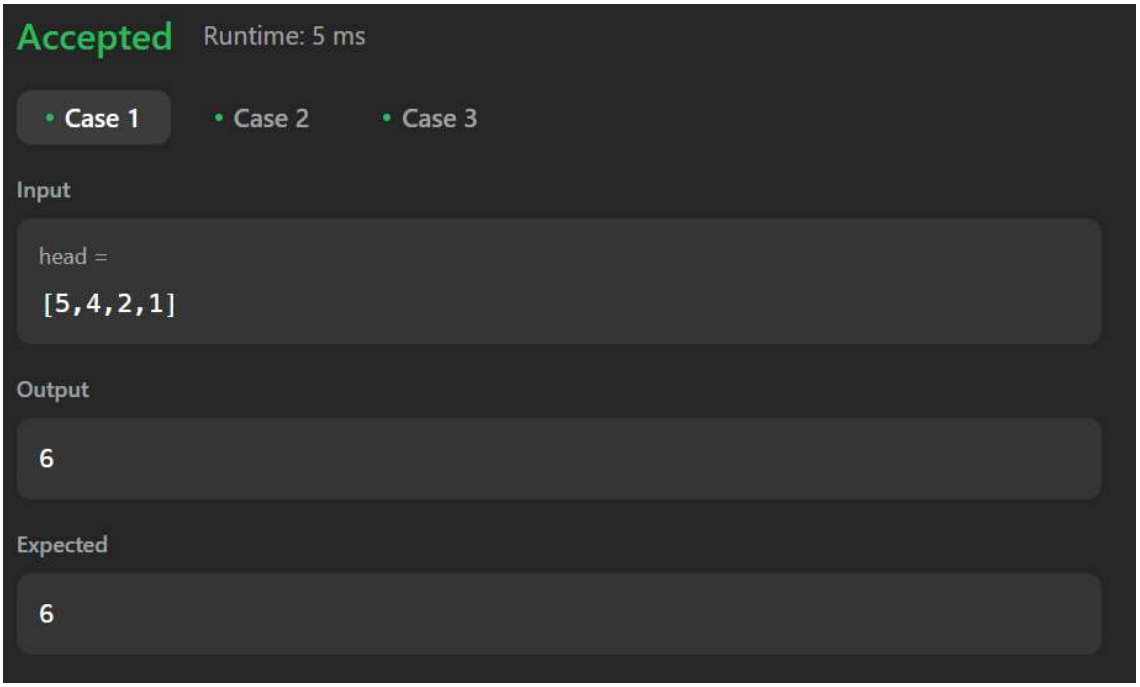
Maximum Twin Sum of a Linked List

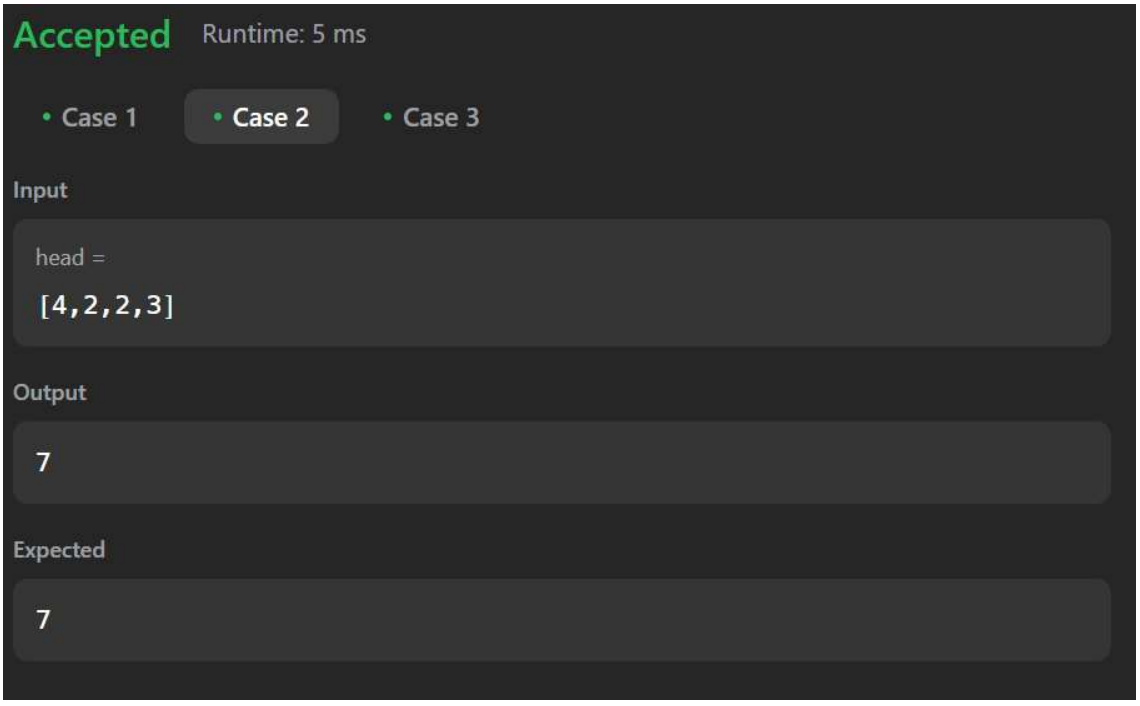
```
/**
* Definition for singly-linked list.
* struct ListNode {
*     int val;
*     struct ListNode *next;
* };
*/
int pairSum(struct ListNode* head) {

    struct ListNode *temp = NULL;
    struct ListNode *q = NULL;
    struct ListNode *p = head;

    // Copying given linked into another linked list.
    while(p!=NULL){
        struct ListNode *newnode = (struct ListNode *)malloc(sizeof(struct ListNode));
        newnode->val = p->val;
        newnode->next = NULL;

        if(temp == NULL){
            temp = newnode;
            q = newnode;

        }
        else{
            temp->next = newnode;
            temp = newnode;
        }
```

```
    p = p->next;
}
p = head;

// Reversing the copied linked list.
struct ListNode *prev = NULL;
while(q!=NULL){

    struct ListNode *next_node = q->next;

    q->next = prev;

    prev = q;

    q = next_node;

}
q = prev;

// Finding the sum of twin nodes using original and reversed linked list.
int maxSum = 0;
while(p!=NULL){
    int sum = p->val + q->val;

    if(sum>maxSum){
        maxSum = sum;
    }
    p = p->next;
    q = q->next;
}
```

```
        return maxSum;
    }
```

Output:





Accepted Runtime: 5 ms

• Case 1 • Case 2 • Case 3

Input

head =

[1,100000]

Output

100001

Expected

100001