WAP to Implement Single Linked List with following operations: Sort the linked list, Reverse the linked list, Concatenation of two linked lists

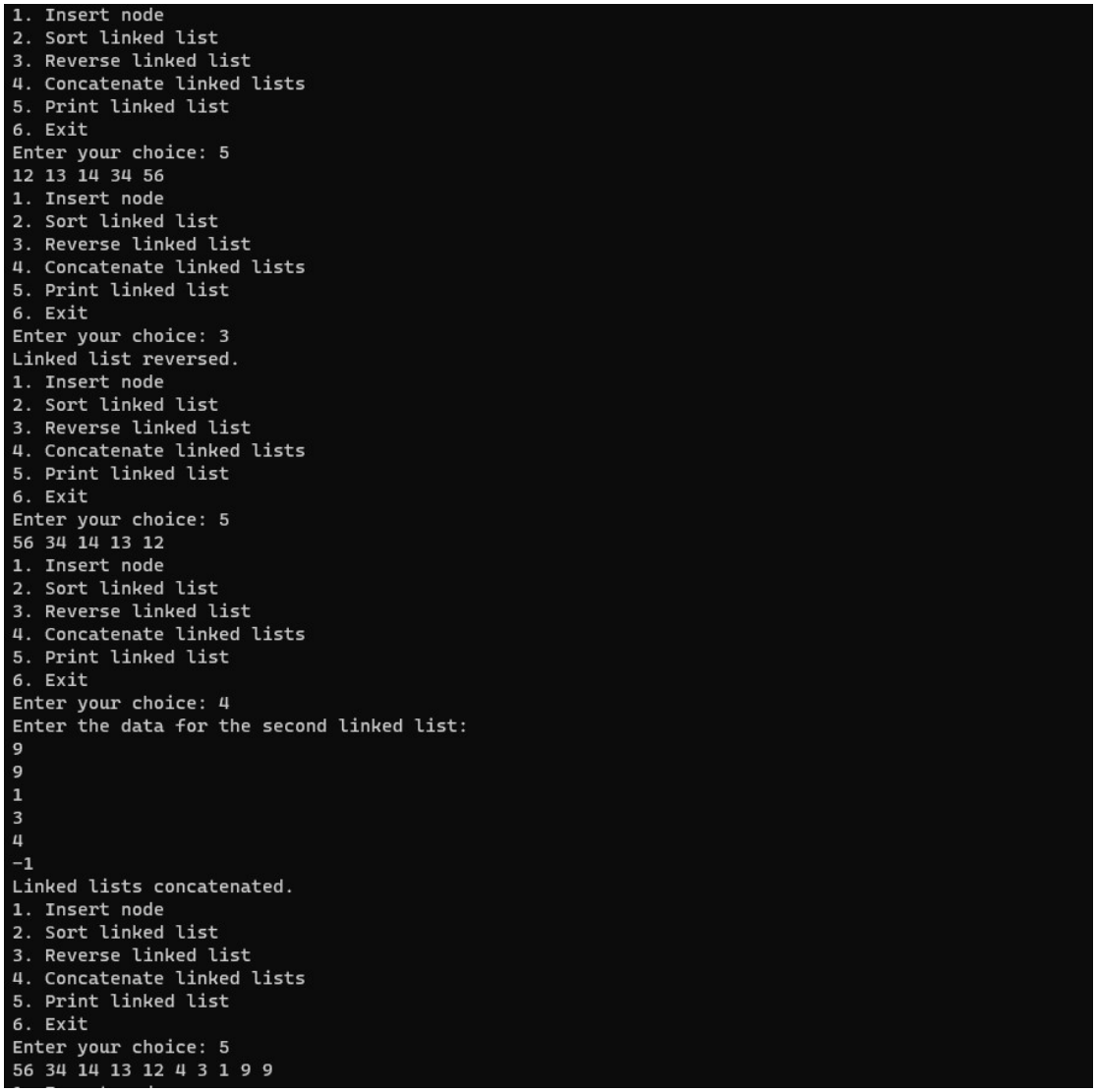
```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *head=NULL;


void enqueue()
{
   struct node *temp;
   temp=(struct node*)malloc(sizeof(struct node));
   printf("Enter value to be inserted: \n");
   scanf("%d",&temp->data);
   temp->next=NULL;
   if(head==NULL)
   {
      head=temp;
      return;
   }
   else
   {
      struct node *ptr;
```

```
        ptr=head;
        while(ptr->next!=NULL)
        {
           ptr=ptr->next;
        }
        ptr->next=temp;
    }
}

void dequeue()
{
    struct node *ptr;
    if(head==NULL)
    {
        printf("Queue is empty \n");
        return;
    }
    else
    {
        ptr=head;
        head=ptr->next;
        printf("Value dequeue=%d",ptr->data);
        free(ptr);
    }
}
```

```
void display()
{
   if(head==NULL)
   {
      printf("Queue is empty \n");
      return;
   }
   else
   {
      struct node *ptr;
      ptr=head;
      while(ptr!=NULL)
      {
         printf("%d \t",ptr->data);
         ptr=ptr->next;
      }
   }
}

void main()
{
   int choice;
   while(1)
   {
      printf("\n 1.to enqueue \n 2.to dequeue\n 3.to display \n 4.exit\n ");
      printf("Enter your choice:");
```

```
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
           enqueue();
           break;
        case 2:
           dequeue();
           break;
        case 3:
           display();
           break;
        case 4:
           exit(0);
           break;
        }
    }
}
```

```
1. Insert node
2. Sort linked list
3. Reverse linked list
4. Concatenate linked lists
5. Print linked list
6. Exit
Enter your choice: 1
Enter the data to insert: 12
1. Insert node
2. Sort linked list
3. Reverse linked list
4. Concatenate linked lists
5. Print linked list
6. Exit
Enter your choice: 1
Enter the data to insert: 34
1. Insert node
2. Sort linked list
3. Reverse linked list
4. Concatenate linked lists
5. Print linked list
6. Exit
Enter your choice: 1
Enter the data to insert: 56
1. Insert node
2. Sort linked list
3. Reverse linked list
4. Concatenate linked lists
5. Print linked list
6. Exit
Enter your choice: 1
Enter the data to insert: 13
1. Insert node
2. Sort linked list
3. Reverse linked list
4. Concatenate linked lists
5. Print linked list
6. Exit
Enter your choice: 1
Enter the data to insert: 14
1. Insert node
2. Sort linked list
3. Reverse linked list
4. Concatenate linked lists
5. Print linked list
6. Exit
Enter your choice: 2
Linked list sorted.
```



WAP to Implement Single Linked List to simulate Stack Operations.

```
#include <stdio.h>

#include <stdlib.h>


struct Node {

   int data;

   struct Node* next;

};
```

```
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void push(struct Node** top, int data) {
    struct Node* newNode = createNode(data);
    newNode->next = *top;
    *top = newNode;
    printf("%d pushed to the stack\n", data);
}

int pop(struct Node** top) {
    if (*top == NULL) {
        printf("Stack is empty\n");
        return -1;
    }

    int poppedValue = (*top)->data;
    struct Node* temp = *top;
    *top = (*top)->next;
    free(temp);

    return poppedValue;
```

```
}

void displayStack(struct Node* top) {
    if (top == NULL) {
        printf("Stack is empty\n");
        return;
    }

    printf("Stack elements: ");
    while (top != NULL) {
        printf("%d ", top->data);
        top = top->next;
    }
    printf("\n");
}

int main() {
    struct Node* stackTop = NULL;
    int choice, data;

    while (1) {
        printf("1. Push\n2. Pop\n3. Display Stack\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
```

```
            case 1:
                printf("Enter data to push: ");
                scanf("%d", &data);
                push(&stackTop, data);
                break;
            case 2:
                printf("Popped element: %d\n", pop(&stackTop));
                break;
            case 3:
                displayStack(stackTop);
                break;
            case 4:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }

    return 0;
}
```

```
1. Push
2. Pop
3. Display Stack
4. Exit
Enter your choice: 1
Enter data to push: 14
14 pushed to the stack
1. Push
2. Pop
3. Display Stack
4. Exit
Enter your choice: 1
Enter data to push: 13
13 pushed to the stack
1. Push
2. Pop
3. Display Stack
4. Exit
Enter your choice: 1
Enter data to push: 15
15 pushed to the stack
1. Push
2. Pop
3. Display Stack
4. Exit
Enter your choice: 1
Enter data to push: 16
16 pushed to the stack
1. Push
2. Pop
3. Display Stack
4. Exit
Enter your choice: 1
Enter data to push: 17
17 pushed to the stack
1. Push
2. Pop
3. Display Stack
4. Exit
Enter your choice: 3
Stack elements: 17 16 15 13 14
1. Push
2. Pop
3. Display Stack
4. Exit
Enter your choice: 2
Popped element: 17
```

```
1. Push
2. Pop
3. Display Stack
4. Exit
Enter your choice: 3
Stack elements: 16 15 13 14
1. Push
2. Pop
3. Display Stack
4. Exit
Enter your choice: 2
Popped element: 16
1. Push
2. Pop
3. Display Stack
4. Exit
Enter your choice: 3
Stack elements: 15 13 14
```

WAP to Implement Single Linked List to simulate Queue Operations.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *head=NULL;

void enqueue()
{
   struct node *temp;
   temp=(struct node*)malloc(sizeof(struct node));
   printf("Enter value to be inserted: \n");
   scanf("%d",&temp->data);
   temp->next=NULL;
```

```
    if(head==NULL)
    {
        head=temp;
        return;
    }
    else
    {
        struct node *ptr;
        ptr=head;
        while(ptr->next!=NULL)
        {
            ptr=ptr->next;
        }
        ptr->next=temp;
    }
}

void dequeue()
{
    struct node *ptr;
    if(head==NULL)
    {
        printf("Queue is empty \n");
        return;
    }
    else
    {
        ptr=head;
```

```
        head=ptr->next;
        printf("Value dequeue=%d",ptr->data);
        free(ptr);
    }
}

void display()
{
    if(head==NULL)
    {
        printf("Queue is empty \n");
        return;
    }
    else
    {
        struct node *ptr;
        ptr=head;
        while(ptr!=NULL)
        {
            printf("%d \t",ptr->data);
            ptr=ptr->next;
        }
    }
}

void main()
{
    int choice;
```

```
    while(1)
    {
        printf("\n 1.enqueue \n 2.dequeue\n 3.display \n 4.exit\n ");
        printf("Enter your choice:");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
            enqueue();
            break;
        case 2:
            dequeue();
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
            break;
        }
    }
}
```

```
 1.enqueue
 2.dequeue
 3.display
 4.exit
 Enter your choice:1
Enter value to be inserted:
13

 1.enqueue
 2.dequeue
 3.display
 4.exit
 Enter your choice:1
Enter value to be inserted:
14

 1.enqueue
 2.dequeue
 3.display
 4.exit
 Enter your choice:1
Enter value to be inserted:
15

 1.enqueue
 2.dequeue
 3.display
 4.exit
 Enter your choice:1
Enter value to be inserted:
16

 1.enqueue
 2.dequeue
 3.display
 4.exit
 Enter your choice:3
13      14      15      16
 1.enqueue
 2.dequeue
 3.display
 4.exit
 Enter your choice:2
Value dequeue=13
 1.enqueue
 2.dequeue
 3.display
 4.exit
 Enter your choice:3
14      15      16
```