Date: 29/01/2024

1) Singly Linked List, sort, reverse, concatination

```
#
Struct node {
        int data;
        Struct node *next;
    }

Struct node *head1 = NULL;
Struct node *head2 = NULL;


Struct node *insert1 (struct node *head, int num) {
        Struct node *new_node, *ptr;
        new_node = (Struct node*) malloc (sizeof (struct node));
        new_node -> data = num;
        new_node -> next = Null;
        if (head1 == NULL) {
            head1 = new_node;
        } else {
            ptr = head1;
            while (ptr -> next != NULL) {
                ptr = ptr -> next;
            }
            ptr -> next = new_node;
        }
        return head1
    }
```

```
Struct node *insert2(struct node *head2, int num){
    Struct node *new_node, *ptr;
    new_node = (struct node*) malloc (sizeof(struct node));
    new_node -> data = num;
    new_node -> next = NULL;
    if (head2 == NULL){
        head2 = new_node;}
    } else{
        ptr = head2;
        while(ptr -> next != NULL) {
                ptr = ptr -> next
        }
        ptr -> next = new_node;
    }
    return head2;
}

void concat (struct node *head1, struct node*head2){
    struct node *ptr;
    if ( head1 != NULL && head2 != NULL){
        ptr = head1.
        while (ptr -> next != NULL){
            ptr = ptr -> next;
        }
        ptr -> next = head2;
    }else{
        printf("either of the Listed List is empty");
}}
```

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

void sort (struct node *head){
    struct node *ptr1, *ptr2;
    int temp;
    ptr1 = head
    while (ptr->next != NULL){
        ptr2 = ptr1->next;
        while (ptr2 != NULL){
            if (ptr1->data > ptr2->data){
                temp = ptr1->data;
                ptr1->data = ptr2->data;
                ptr2->data = temp;
            }
            ptr2 = ptr2->next;
        }
        ptr1 = ptr1->next;
    }
} display (head);
```

```
void reverse (struct node *head)
{
    struct node *ptr, *prev;
    ptr = head;
}
```

```
void reverse (struct node *head){
    struct node *ptr = NULL, *prev = NULL;
    ptr = head
    while (head != NULL){
        ptr = head->next;
        head->next = prev;
        prev = head;
        head = ptr;
    }
    head = prev;
}
display(head);
}
}
```

---

o/p:

Enter your choice : 1

Enter the number : 10

Enter your choice : 1

Enter the number : 35

Enter your choice : 1

Enter the number : 24

Enter your choice : 3

10 → 35 → 24 →

Enter your choice: 2

Enter the number: 32

Enter your choice: 2

Enter the number: 21

Enter your choice: 2

Enter the number: 2

Enter your choie: 4

32 → 21 → 2 →

Enter your choie: 5

10 → 35 → 24 → 32 → 21 → 2 →

Enter your choice: 6

2 → 10 → 21 → 24 → 32 → 35

Enter your choice: 7

35 → 32 → 24 → 21 → 10 → 2 →

3a. Stack implementation using single Linked List

```
struct node {
    int data;
    struct node *next;
};

struct node *head = NULL;

void push() {
    struct node *new_node;
    int num;
```

```
printf("Enter the number");
scanf("%d",&num);

new_node = (struct node*) malloc(sizeof(struct node));
new_node->data = num;

if(head == NULL){
    head = new_node;
    new_node->next = NULL;
}else{
    new_node->next = head;
    head = new_node;
}
}

void pop(){
    struct node *ptr;
    if(head == NULL){
        printf("Underflow...\n");
    }else{
        ptr=head;
        head = ptr->next;
        printf("Poped element : %d \n", ptr->data);
        free(ptr);
    }
}
```

```
void display()
{
    struct node *ptr;
    ptr = head;
    printf("Elements of Linked List: ");
    while(ptr != NULL){
        printf("%d -> ", ptr->data);
        ptr = ptr -> data;
    }
    printf("\n");
}
```

O/P:

Enter Choice : 1

Enter the number : 10

Enter your choice : 1

Enter the number : 20

Enter your choice : 1

Entery the number : 30

Enter your choice :

Element of Linked List : 30 → 20 → 10 →

Enter the number : 2

poped element : 30

Enter the number : 2

poped element : 20

Enter the number : 2

poped element : 10

Sun. 5/2/24

Enter the number : 2

Underflow...

2b) Queue Implementation using Single Linked List

```
Struct node {
    int data;
    Struct node *next;
};

Struct node *head = NULL;

void enqueue () {
    Struct node *ptr, *new_node;
    int num;

    printf ("Enter the number:");
    Scanf ("%d", &num);

    new_node = (Struct node*) malloc (sizeof ( struct node
    new_node->data = num;
    new_node->next = NULL;

    if (head == NULL){
      head = new_node;
    }else {
        ptr = head;
        while (ptr-> next != NULL)
        {
          ptr = ptr->next;
        }
        ptr->next = new_node
};
```

```
void dequeue() {
    struct node *ptr;
    if (head == NULL) {
        printf("Under Flow...");
    else {
        ptr = head;
        head = ptr->next;
        printf("Poped element: %d \n", ptr->data);
        free(ptr);
    }
}

void display() {
    Struct node *ptr;
    if (head == NULL) {
        printf("Queue is Empty ... \n");
    else {
        ptr = head;
        printf("Elements of Linked List : \n");
        while (ptr != NULL) {
            printf("%d", ptr->data);
            ptr = ptr->next;
        }
        printf("\n");
    }
}
```

P.T.O

o/p:

```
Enter your choice : 1
number : 10
choice : 1
number : 20
choice : 1
number : 30
choice : 3
Elements of Linked List:
10 → 20 → 30 →
choice : 2
dequeue element : 10
choice : 2
dequeue element : 20
choice : 2
dequeue element : 30
choice : 2
queue is Empty...
```

Sun
5/2/24