24/11/25

6)

a) WAP to implement single link list with following operations: sort the linked list, reverse the linked list, concatenation of two linked list

Pseudo code:

```
Void bubblesort() {
    for (i=head; i->next!=NULL; i=i->next) {
        for (j=i->next; j!=NULL; j=j->next) {
            if (i->data > j->data) {
                tempData = i->data;
                i->data = j->data;
                j->data = tempdata;
            }
        }
    }
```

```
struct node
Void reverseLinkedList() {
    while (curr != NULL) {
        next = curr->next;
        curr->next = prev;
        Prev = curr;
        curr = next;
    }
    return prev
}
```

```
Void concatenation() {
    LinkedList
    if (head2 == NULL) return head1;
    temp = head1;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = head2
    return head1;
}
```

code:-

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
};
struct node *head=NULL, *head1=NULL, *head2=NULL;
Void createList (struct node **head, int n) {
    struct node *newnode, *temp;
    int data, i;
    if (n<0) {
        printf ("n should be greater than 0");
        return;
    }
    for (i=1; i<=n; i++) {
        newnode = (struct node*) malloc (sizeof
                (struct node));
        if (newnode == NULL) {
            Printf ("memory allocation failed");
            return;
        }
        Printf ("Enter data: ");
        Scanf ("%d", &data);
        newnode -> data = data;
        newnode -> next = NULL;
        if (*head == NULL) {
            *head = newnode;
        }
        else {
            temp -> next = newnode;
        }
        temp = newnode;
    }
    Printf ("Linked list is created.");
}
```

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

void bubblesort(struct node *head){
    if(head==NULL||head->next==NULL)return;
    struct node *i,*j;
    int tempdata;
    for(i=head;i->next!=NULL;i=i->next){
        for(j=i->next;j!=NULL;j=j->next){
            if(i->data>j->data){
                tempdata=i->data;
                i->data=j->data;
                j->data=tempdata;
            }
        }
    }
    printf("\nLinked list sorted successfully!!");
}
```

```
struct node *reverseLinkedList(struct node *head){
    struct node *prev=NULL,*next=NULL,*curr=head;
    while(curr!=NULL){
        next=curr->next;
        curr->next=prev;
        prev=curr;
        curr=next;
    }
    printf("Linked list reversed successfully!\n");
    return prev;
}

struct node *concat(struct node *head1,struct node *head2){
    struct node *temp;
    if(head2==NULL){
        return head1;
    }
    if(head1==NULL){
        return head2;
    }
    temp=head1;
    while(temp->next!=NULL){
        temp=temp->next;
    }
    temp->next=head2;
    printf("Linked Lists concatenated successfully");
    return head1;
}
```

```
void main(){
  int n, ch, listchoice;
  do{
    printf("\n1)Create LL 2)Sort 3)Reverse the
        LL 4)Concat LL's 5)Display LL");
    printf("\nEnter -1 to exit");
    printf("\n Enter your choice:");
    scanf("%d",&ch);
    switch(ch){
      case 1:
        printf("\nEnter the List choice(1or2):");
        scanf("%d",&listchoice);
        printf("\n Enter no. of nodes:");
        scanf("%d",&n);
        if(listchoice==1){
            createList(&head1,n);
        }elseif(listchoice==2){
            createList(&head2,n);
        }else{
            Printf("\nInvalid choice");
        }
        break;
case 2:
    printf("\n Enter the listchoice(1or2):");
    scanf("%d",&listchoice);
    if(listchoice==1){
        bubblpsort(head1);
    }elseif(listchoice==2){
        bubblesort(head2);
    }else{
        Printf("\nInvalid Choice");
    }break;
```

```
    Printf("\n Enter the listchoice(1or2):");
    Scanf("%d",&listchoice);
    if(listchoice==1){
        head1=ReverseLinkedList(head1);
    }else if(listchoice==2){
        head2=ReverseLinkedList(head2);
    }else{
        printf("\nInvalid choice");
    }
    break;
case4;
    head1=concat(head1,head2);
    break;
case 5;
    PrintfC"\nList 1\n");
    display(head1);
    printf("\nList 2\n");
    display(head2);
    break;
default: printf("Invalid input.");
  }
}while(ch!=-1);
}
1) Create LL 2) Sort LL 3) Reverse LL 4) Concat LL's
5) Display LL
enter -1 to exit
Enter your choice:1
Enter the listChoice(1or2):1
```

```
Enter no. of nodes: 5
enter data: 2
Enter data: 4
enter data: 3
enter data: 1
enter data: 5
Linked List is created.

1) create LL 2) sort LL 3) Reverse LL 4) concat LL's 5) Display LL
Enter -1 to exit
Enter your choice: 1

Enter 1st choice (1 or 2): 2

Enter no. of nodes: 5
enter data: 7
enter data: 6
Enter data: 8
enter data: 9
enter data: 10
Linked list is created.

1) Create LL 2) sort LL 3) Reverse LL 4) concat LL's
5) Display LL
Enter -1 to exit
Enter your choice: 5
List 1
Linked List:
2 4 3 1 5
List 2
Linked List:
7 6 8 9 10

1) Create LL 2) sort LL 3) Reverse LL 4) concat LL's
5) Display LL
Enter -1 to exit
Enter your choice: 2
Enter the 1st choice (1 or 2): 1
Linked list sorted successfully!!

1) Create LL 2) sort LL 3) Reverse LL 4) concat LL's
5) Display LL
Enter -1 to exit
Enter your choice: 3
Enter the 1st choice (1 or 2): 2
Linked List reversed successfully!.

1) Create LL 2) sort LL 3) Reverse LL 4) concat LL's
5) Display LL
Enter -1 to exit
Enter your choice: 4
Linked lists concatenated successfully!

1) Create LL 2) sort LL 3) Reverse LL 4) concat LL's
5) Display LL
Enter -1 to exit
Enter your choice: 5
List 1
Linked list:
1 2 3 4 5 10 9 8 6 7
List 2
Linked list:
10 9 8 6 7
```