Program 6a

Write A Program to Implement Single Link List with following operations:
Sort the linked list,
Reverse the linked list,
Concatenation of two linked lists.

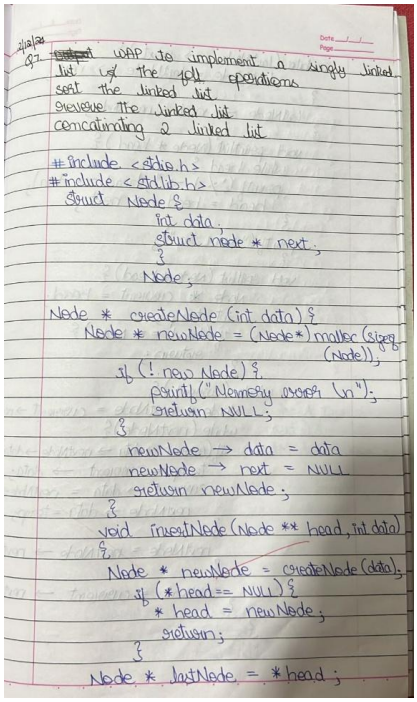
Code:

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

typedef struct Node {
    int data;
    struct Node* next;
} Node;

Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory error\n");
        return NULL;
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void insertNode(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* lastNode = *head;
    while (lastNode->next) {
        lastNode = lastNode->next;
    }
    lastNode->next = newNode;
}

void printlist(Node* head) {
    Node* current = head;
    while (current) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}
```

```
void sortlist(Node* head) {
    if (head == NULL) {
        return;
    }
    Node* current;
    Node* nextNode;
    int temp;
    for (current = head; current != NULL; current = current->next) {
        for (nextNode = current->next; nextNode != NULL; nextNode = nextNode->next) {
            if (current->data > nextNode->data) {
                temp = current->data;
                current->data = nextNode->data;
                nextNode->data = temp;
            }
        }
    }
}

void reverselist(Node** head) {
    Node* prev = NULL;
    Node* current = *head;
    Node* next = NULL;
    while (current) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    *head = prev;
}

void concatenate(Node** head1, Node* head2) {
    if (*head1 == NULL) {
        *head1 = head2;
        return;
    }
    Node* lastNode = *head1;
    while (lastNode->next) {
        lastNode = lastNode->next;
    }
    lastNode->next = head2;
}

int main() {
    Node* head1 = NULL;
    Node* head2 = NULL;

    insertNode(&head1, 1);
    insertNode(&head1, 3);
    insertNode(&head1, 5);
```

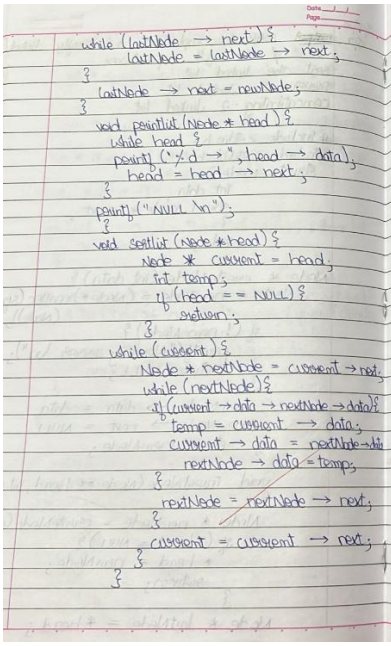
```
    insertNode(&head2, 2);
    insertNode(&head2, 4);
    insertNode(&head2, 6);

    printf("Linked list 1: ");
    printlist(head1);

    printf("Linked list 2: ");
    printlist(head2);

    sortlist(head1);
    printf("Sorted Linked list 1: ");
    printlist(head1);

    reverselist(&head2);
    printf("Reversed Linked list 2: ");
    printlist(head2);

    concatenate(&head1, head2);
    printf("Concatenated Linked list: ");
    printlist(head1);

    return 0;
}
```

```
Linked list 1: 1 -> 3 -> 5 -> NULL
Linked list 2: 2 -> 4 -> 6 -> NULL
Sorted Linked list 1: 1 -> 3 -> 5 -> NULL
Reversed Linked list 2: 6 -> 4 -> 2 -> NULL
Concatenated Linked list: 1 -> 3 -> 5 -> 6 -> 4 -> 2 -> NULL
```

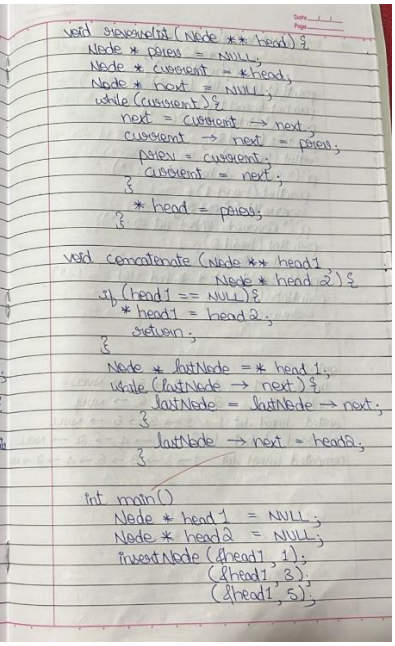


WAP to implement a singly linked list. Use the following operations
sort the linked list
reverse the linked list
concatinating 2 linked list

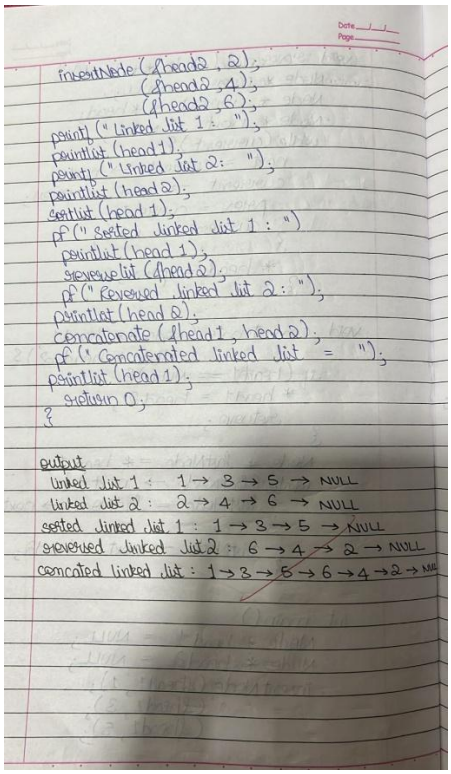
```
#include <stdio.h>
#include <stdlib.h>
struct Node {
        int data;
        struct node * next;
        };
        Node;

Node * createNode (int data) {
    Node * newNode = (Node*) malloc (sizeof
                                (Node));
        if (! newNode) {
            printf("Memory error\n");
            return NULL;
        }
        newNode -> data = data
        newNode -> next = NULL
        return newNode;
    }
    void insertNode (Node ** head, int data)
    {
        Node * newNode = createNode (data);
        if (*head== NULL) {
            * head = newNode;
            return;
        }
    Node * lastNode = *head;
```



```
while (lastNode -> next) {
        lastNode = lastNode -> next;
}
lastNode -> next = newNode;
}
void printlist (Node * head) {
while head {
printf ("%d -> ", head -> data);
head = head -> next;
}
printf ("NULL \n");
}
void sortlist (Node *head) {
    Node * current = head;
    int temp;
    if (head == NULL) {
        return;
    }
while (current) {
    Node * nextNode = current -> next;
    while (nextNode){
      if (current -> data -> nextNode -> data){
        temp = current -> data;
        current -> data = nextNode -> da
        nextNode -> data = temp;
      }
      nextNode = nextNode -> next;
    }
    current = current -> next;
}
}
```



```
void reverselist (Node ** head) {
    Node * prev = NULL;
    Node * current = *head;
    Node * next = NULL;
    while (current) {
        next = current -> next;
        current -> next = prev;
        prev = current;
        current = next;
    }
    * head = prev;
}

void concatenate (Node ** head1,
                  Node * head2) {
    if (head1 == NULL) {
        * head1 = head2;
        return;
    }
    Node * lastNode = * head1;
    while (lastNode -> next) {
        lastNode = lastNode -> next;
    }
    lastNode -> next = head2;
}

int main()
{
    Node * head1 = NULL;
    Node * head2 = NULL;
    insertNode (&head1, 1);
               (&head1, 3);
               (&head1, 5);
```



```
insertNode (&head2, 2);
           (&head2, 4);
           (&head2, 6);
printf (" Linked list 1 : ");
printlist (head1);
printf (" Linked list 2: ");
printlist (head2);
sortlist (head1);
pf (" Sorted linked list 1 : ")
 printlist (head1);
 reverselist (&head2);
 pf (" Reversed linked list 2 : ");
printlist (head2);
concatenate (&head1, head2);
pf (" Concatenated linked list = ");
printlist (head1);
 return 0;
}
```

output

Linked list 1 : 1 → 3 → 5 → NULL
Linked list 2 : 2 → 4 → 6 → NULL
sorted linked list 1 : 1 → 3 → 5 → NULL
reversed linked list 2 : 6 → 4 → 2 → NULL
concated linked list : 1 → 3 → 5 → 6 → 4 → 2 → NULL