Program 7

a) Write a Program to Implement Single Link List with following operations:

(i)Sort the linked list,

(ii) Reverse the linked list,

(iii)Concatenation of two linked lists.

Observation:

WEEK-8 2/12/2024.

a)

```
Void reverseList (struct Node** head){
    struct Node *prev = NULL, *current = *head,
      * next = NULL;
    while (current != NULL){
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    *head = prev;
}

Void sortList (struct Node* head){
    struct Node *i, *j;
    int temp;
    for (i = head; i != NULL; i = i->next){
        for (j = i->next; j != NULL; j = j->next){
            if (i->data > j->data){
                temp = i->data;
                i->data = j->data;
                j->data = temp;
            }
        }
    }
}
```

```
void concatenateList (struct Node **head1, struct Node **head2){
    if (*head1 == NULL){
        *head1 = head2;
    } else {
        struct Node *temp = *head1;
        while (temp->next != NULL){
            temp = temp->next;
        }
        temp->next = head2;
    }
}
```

b)Write a Program to Implement Single Link List to simulate
(i)Stack
(ii)Queue Operations.

Observation:

b) WAP to implement Single Link List to stimulate stack & Queue Operations.

```
struct Node *top = NULL;
void push (int x) {
    struct Node *newNode = (struct Node*) malloc(sizeof (struct Node));
    newNode->data = x;
    newNode->next = top;
    top = newNode;
}
void pop () {
    struct Node *temp;
    temp = top;
    if (top == NULL) {
        printf ("stack Underflow\n");
}
```

```
else {
    printf("Popped element is %d\n", top->data);
        top = top->next;
        free(temp);
    }
}
```

- Queue Operations.

```
struct Node* front = NULL;
struct Node* rear = NULL;

~~void (int x)~~
void enqueue(int x) {
    struct Node *newNode = (struct Node*)malloc(sizeof(struct
        newNode->data = x;
        newNode->next = NULL;
    if (front == NULL && rear == NULL) {
        front = rear = newNode;
    }
    else {
        rear->next = newNode;
        rear = newNode;
    }
}
```

```
void dequeue() {
    struct Node *temp;
    temp = front;
    if (front == NULL && rear == NULL) {
        printf("Queue is empty\n");
    } else if (front == rear) {
        front = rear = NULL;
    } else {
        front = front->next;
        free(temp);
    }
}
```

23/12/24

Code:

```
#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 insertEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        struct Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}


void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}


void sortList(struct Node* head) {
    struct Node *i, *j;
    int temp;
    for (i = head; i != NULL; i = i->next) {
        for (j = i->next; j != NULL; j = j->next) {
            if (i->data > j->data) {
                temp = i->data;
```

```
                i->data = j->data;
                j->data = temp;
            }
        }
    }
}


void reverseList(struct Node** head) {
    struct Node *prev = NULL, *current = *head, *next = NULL;
    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
   *head=prev;
}


void concatenateLists(struct Node** head1, struct Node* head2) {
    if (*head1 == NULL) {
        *head1 = head2;
    } else {
        struct Node* temp = *head1;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = head2;
    }
}


int main() {
    struct Node* list1 = NULL;
    struct Node* list2 = NULL;


    insertEnd(&list1, 5);
    insertEnd(&list1, 1);
    insertEnd(&list1, 9);
    insertEnd(&list1, 3);


    insertEnd(&list2, 8);
    insertEnd(&list2, 2);
    insertEnd(&list2, 4);


    printf("List 1: ");
    printList(list1);
```

```c
    printf("List 2: ");
    printList(list2);

    sortList(list1);
    printf("\nList 1 after sorting: ");
    printList(list1);


    reverseList(&list2);
    printf("\nList 2 after reversing: ");
    printList(list2);


    concatenateLists(&list1, list2);
    printf("\nList 1 after concatenation with List 2: ");
    printList(list1);

    return 0;
}
```

Output:

```
PS C:\Users\satis\OneDrive\Desktop>  & 'c:\Users\satis\.vscode\extensions\ms-vscode.cpptools-1.22.11-win32-x64\debugAdapters\bin\W
indowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-rktevcfs.pxp' '--stdout=Microsoft-MIEngine-Out-fqep0jvm.znr' '--stderr=Mic
rosoft-MIEngine-Error-u5zpearj.1nu' '--pid=Microsoft-MIEngine-Pid-l15zr4xd.ahj' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpr
eter=mi'
List 1: 5 -> 1 -> 9 -> 3 -> NULL
List 2: 8 -> 2 -> 4 -> NULL

List 1 after sorting: 1 -> 3 -> 5 -> 9 -> NULL

List 2 after reversing: 4 -> 2 -> 8 -> NULL

List 1 after concatenation with List 2: 1 -> 3 -> 5 -> 9 -> 4 -> 2 -> 8 -> NULL
```