program 8: Linked list → sort → concat → reverse

Date
Page

a) Sorting
b) concat of two linked list
c) reverse of linked list

pseudocode:

```
void sortList(){
    if (head == NULL),
        return;
    for(current=head; current->next!=NULL; current=current->next)
    for(index=current->next; index!=NULL; index=index->next)
    if (current->data > index->data){
        temp= current -> data;
        current->data = index->data
        index->data = temp; }}}

void concat(){
    struct Node* temp = list1;
    while (temp->next != NULL) temp = temp->next
    temp->next = list2;
    return list1; }

void reverse(){ while (current != NULL){
    nextNode = current->next
    current->next= prev;
    prev = current;
    current = nextNode;
}
head
return prev;
}
```

code:

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

Struct Node{
    int data;
    Struct Node *next;}

Struct Node *head = NULL
void sort() {
    if (head == NULL) return;

    int swap;
    Struct Node* ptr;
    struct Node* lptr = NULL;
    do{
        swap = 0
        ptr = head
    while (ptr -> next != lptr){
        if (ptr -> data > ptr -> next -> data){
            int temp = ptr -> data;
            ptr -> data = ptr -> next -> data;
            ptr -> next -> data = temp;
            swap = 1;
        }
        ptr = ptr -> next;
    }
    lptr = ptr;
} while (swap);
printf("List sorted.\n");
}
```

```
void reverse() {
    struct Node *prev = NULL, * curr = head, *next = NULL;
    while (curr != NULL){
        next = curr -> next;
        curr -> next = prev;
        prev = curr;
        curr = next;
    }
    head = prev;
    printf("List reversed \n");
}

void concat(){
    int n, data;
    struct Node *newNode, *head2 = NULL,
        *temp = NULL;

    printf("Enter number of nodes for second
        list:");
    scanf("%d", &n);

    if (n<=0){
        printf("second list is empty. Nothing
            to concat. \n");
        return;
    }

    for (int i=1; i<=n; i++){
        newNode = (struct Node*) malloc(sizeof(struct
            Node));
        printf("Enter data for node %d: ", i);
        scanf("%d", &data);
        newNode -> data = data;
```

```
    newNode->next = NULL;
    if(head2 == NULL)
        head2 = newNode;
    else
        temp2 -> next = newNode;
    temp2 = newNode;
}

    if(head == NULL){
    head = head2;
    printf("Lists concatenated.\n");
    return;
    }
    struct Node *temp = head;
    while(temp->next != NULL)
        temp= temp -> next;
    temp->next = head2;

    printf("List concatenated\n");
}
int main(){
    int choice, n;
    printf("1.SORTING");
    printf("2.REVERSING");
    printf("3.Concatenation");
    printf("4.Display");
    printf("5.exit");
    while(1) {
printf("enter your choice:");
scanf("%d", &choice);
switch(choice){
    case 1: sort(); break;
    case 2: reverse(); break;
    case 3: concat(); break;
```

Date
Page

```
case 4 : display(); breat;
case 5 : exit(); break;
default : printf("wrong choice")}} ;
```

output: ---Singly linked list operation---

1. Create
2. Sorting
3. Reversing
4. Concatenation
5. Display
6. Exit

Enter your choice : 1
Enter number of node : 3
Enter data for node1: 1
Enter data for node2 : 2
Enter data for node3: 3
Enter your choice : 3
Enter your choice : 5
Linked list : 3 → 2 → 1 → NULL
Enter your choice: 4
Enter number of nodes for second list: 3
Enter data for node1 : 6
Enter data for node 2 : 4
Enter data for node 3 : 5
Enter your choice : 5
Linked list : 3 → 2 → 1 → 6 → 4 → 5 → NULL
Enter your choice : 2
Enter your choice : 5
Linked list : 1 → 2 → 3 → 4 → 5 → 6 → NULL
Enter your choice : 6
Exiting....

program 8 & 9: singly linked list

a) Stimulate stack

b) Stimulate queue

pseudocode: void ~~implementStack~~ push(item) {

```
struct Node * new_node = (struct Node *) malloc
                         (sizeof(structNode))
new_node -> data = new_data;
new_node -> next = (* head_ref);
(* head_ref) = new_node;
```

}

```
void pop()
{   ptr = head
    head = ptr -> next;
    free(ptr);
```

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

```
void Dequeue () {
    ptr = head;
    head = ptr -> next;
    free(ptr);
}
```

Date
Page

code (a)

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

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

struct Node *head=NULL;

void push(struct Node** head_ref, int new_data){
    struct Node* newNode = (struct Node*)
                malloc(sizeof(struct Node));
    newNode->data=new_data;
    new_node->next=(*head_ref);
    (*head_ref)=newNode;
}

void pop(){
    struct Node * temp;
    if(head==NULL){
        printf("\nList is empty\n");
    }else{
        temp=head;
        head=temp->next;
        free(temp);
        printf("Node deleted from begining...\n");
    }
}

int main(){
    int choice;
    int element;
```

```
printf("\n 1.Push \n 2.Pop \n 3. Display \n 4. Exit");
while (1){
    printf ("Enter your choice: ");
    scanf ("%d", &choice);
    switch (choice){
        case 1: printf ("Enter an element to push: ");
                scanf ("%d", &element);
                push (&head, element);
                break;
        case 2: pop();
                break;
        case 3: displayList();
                break;
        case 4: exit (0);
        default: printf("Wrong choice! Please try again!");
    }
}
}
```

output: 1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter an element to push: 1
Enter your choice: 2
Node deleted.
Enter your choice: 2
Stack is empty
Enter your choice 1
Enter an element to push: 1
Enter your choice 1
Enter an elemment to push: 2

Enter your choice 1
Enter element to push: 3
Enter your choice 3
Stack: 3 -> 2 -> 1 -> NULL
Enter your choice : 9
Exiting....

(b)
```
#include<stdio.h>
#include<stdlib.h>

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

struct Node *head = NULL;

void Enqueue(int item) {
    struct Node * ptr, * temp;
    ptr = (struct Node*) malloc (sizeof(struct Node
    ptr->data=item;
    ptr->next = NULL;

    if(head == NULL){
        head = ptr;
        printf("\n Node inserted\n");
    }else{
        temp= head;
        while(temp->next != NULL){
            temp = temp->next;
        }
        temp->next=ptr;
```

```
        printf("Node inserted \n");
    }
}
void Dequeue(){
    struct Node *ptr;
    if(head == NULL){
        printf("Queue is empty");
    }else{
        ptr = head;
        head = ptr->next;
        free(ptr);
    }
}

int main(){
    int x;
    int element;
    printf("\n1: Enqueue 2: Dequeue 3: Display \n");
    while(1){
        printf("Enter your choice: ");
        scanf("%d", &x);
        switch(x){
            case1:
                printf("Enter an element to enque:");
                scanf("%d", &element);
                Enqueue(element);
                break;

            case 2: Dequeue();
                    break;
            case 3: displayList();
                    break;
```

```
        case 4: printf("Exiting...\n");
                exit(0);
        default: printf("Wrong choice! Try again");
                break;
    }
}
}
```

output: 1: Enqueue
2: Dequeue
3: Display
4: Exit
Enter your choice 1
Enter an element to enqueue 1
Enter your choice 2
Node deleted
Enter your choice 2
Queue empty
Enter your choice 1
Enter an element to enqueue 1
Enter your choice 1
Enter an element to enqueue 2
Enter your choice 1
Enter an element to enqueue 3
Enter your choice 3
1 → 2 → 3 → NULL
Enter your choice 2
node deleted
Enter your choice 3
2 → 3 → NULL
Enter your choice 4
Exiting...

O/P seen