CS102 - Lab 9 - 14/03/2024

```c
//SHIVAMBU DEV PANDEY
//23BCS123
//CSE B
//LAB 9
//PROGRAM 1
//Date : 14/03/2024

/*
1.Write a C program to merge two singly linked lists and return
the head of the merged linked list.
List 1 has the nodes: 4->1->2
List 2 has the nodes: 10->7->8
Output (merged list): 4->1->2->10->7->8
*/

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

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

void traverse(node *head) {
   int count = 1;
   node *p = head;
   while (p != NULL) {
       printf("\nNode_%d _data: %d", count, p->data);
       count++;
       p = p->next;
   }
}

void insert_end(node **head, int item) {
   node *new_node = (node*) malloc(sizeof(node));
   new_node->data = item;
   new_node->next = NULL;
   if (*head == NULL) {
```

```
        *head = new_node;
        return;
    }
    node *p = *head;
    while(p->next != NULL) {
        p = p->next;
    }
    p->next = new_node;
}

node* merge_ll(node *head1, node *head2) {
    if (head1 == NULL)
        return head2;

    node *temp = head1;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = head2;

    return head1;
}

int main() {
    node *head1 = NULL, *head2 = NULL;


    printf("List 1\n");
    insert_end(&head1, 4);
    insert_end(&head1, 1);
    insert_end(&head1, 2);
    traverse(head1);

    printf("\nList 2\n");
    insert_end(&head2, 10);
    insert_end(&head2, 7);
    insert_end(&head2, 8);
    traverse(head2);

    head1 = merge_ll(head1, head2);
```

```
    printf("\nMerged List:\n");
    traverse(head1);

    return 0;
}
```

OUTPUT:

```
der/23BCS123/23iiit@iiit-OptiPlex-3090:~/Desktop/New Folder/23BCS123
 23BCS123_LAB9_P1.c -o 23BCS123_LAB9_P1 && "/home/iiit/Desktop/New F
List 1
4 -> 1 -> 2 -> (NULL)

List 2
10 -> 7 -> 8 -> (NULL)

Merged List:
4 -> 1 -> 2 -> 10 -> 7 -> 8 -> (NULL)
iiit@iiit-OptiPlex-3090:~/Desktop/New Folder/23BCS123/23BCS123_LAB9$
```

```
//SHIVAMBU DEV PANDEY
```

```c
//23BCS123
//CSE B
//LAB 9
//PROGRAM 2
//Date : 14/03/2024

/*
2.Write a C program to merge two singly linked lists
and return the head of the merged linked list.
List 1 has the nodes: 4->1->2
List 2 has the nodes: 10->7->8
Output (merged list): 4->1->2->8->7->10
*/

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

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

void traverse(node *head) {
   node *p = head;
   while (p != NULL) {
       printf("%d -> ", p->data);
       p = p->next;
   }
   printf("(NULL)\n");
}

void insert_end(node **head, int item) {
   node *new_node = (node*) malloc(sizeof(node));
   new_node->data = item;
   new_node->next = NULL;
   if (*head == NULL) {
       *head = new_node;
       return;
   }
   node *p = *head;
```

```
    while(p->next != NULL) {
        p = p->next;
    }
    p->next = new_node;
}

node* rev_ll(node* head) {
    node* prev = NULL;
    node* current = head;
    node* next = NULL;

    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    return prev;
}


node* merge_ll(node *head1, node *head2) {
    if (head1 == NULL)
        return head2;

    head2 = rev_ll(head2);

    node *temp = head1;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = head2;

    return head1;
}

int main() {
    node *head1 = NULL, *head2 = NULL;

    printf("List 1\n");
```

```
    insert_end(&head1, 4);
    insert_end(&head1, 1);
    insert_end(&head1, 2);
    traverse(head1);

    printf("\nList 2\n");
    insert_end(&head2, 10);
    insert_end(&head2, 7);
    insert_end(&head2, 8);
    traverse(head2);

    head1 = merge_ll(head1, head2);

    printf("\nMerged List:\n");
    traverse(head1);

    return 0;
}
```

OUTPUT:

```
der/23BCS123/23BCS123_LAB9/"23BCS123_LAB9_P2
iiit@iiit-OptiPlex-3090:~/Desktop/New Folder/23BCS123$ cd "/home/iiit/De
P2.c -o 23BCS123_LAB9_P2 && "/home/iiit/Desktop/New Folder/23BCS123/23BC
List 1
4 -> 1 -> 2 -> (NULL)

List 2
10 -> 7 -> 8 -> (NULL)

Merged List:
4 -> 1 -> 2 -> 8 -> 7 -> 10 -> (NULL)
iiit@iiit-OptiPlex-3090:~/Desktop/New Folder/23BCS123/23BCS123_LAB9$
```

```c
//SHIVAMBU DEV PANDEY
//23BCS123
//CSE B
//LAB 9
//PROGRAM 3
//Date : 14/03/2024

/*
3.Write a C program to merge the two given sorted linked lists.
The resulting merged linked list should also be sorted.
List 1 has the nodes: 4->8->15->19
List 2 has the nodes: 7->9->10->16
Output (merged list):
4->7->8->9->10->15->16->19
*/

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

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

void traverse(node *head) {
   node *p = head;
   while (p != NULL) {
       printf("%d -> ", p->data);
       p = p->next;
   }
   printf("(NULL)\n");
}

void insert_end(node **head, int item) {
   node *new_node = (node*) malloc(sizeof(node));
   new_node->data = item;
   new_node->next = NULL;
   if (*head == NULL) {
       *head = new_node;
       return;
```

```
    }
    node *p = *head;
    while(p->next != NULL) {
        p = p->next;
    }
    p->next = new_node;
}

node* merge_sorted_ll(node *head1, node *head2) {
    if (head1 == NULL)
        return head2;
    if (head2 == NULL)
        return head1;

    node *merged_list = NULL;
    node *temp = NULL;

    if (head1->data <= head2->data) {
        merged_list = head1;
        head1 = head1->next;
    } else {
        merged_list = head2;
        head2 = head2->next;
    }
    temp = merged_list;

    while (head1 != NULL && head2 != NULL) {
        if (head1->data <= head2->data) {
            temp->next = head1;
            temp = head1;
            head1 = head1->next;
        } else {
            temp->next = head2;
            temp = head2;
            head2 = head2->next;
        }
    }

    if (head1 != NULL)
        temp->next = head1;
```

```
    if (head2 != NULL)
        temp->next = head2;

    return merged_list;
}

int main() {
    node *head1 = NULL, *head2 = NULL;

    printf("List 1\n");
    insert_end(&head1, 4);
    insert_end(&head1, 8);
    insert_end(&head1, 15);
    insert_end(&head1, 19);
    traverse(head1);

    printf("\nList 2\n");
    insert_end(&head2, 7);
    insert_end(&head2, 9);
    insert_end(&head2, 10);
    insert_end(&head2, 16);
    traverse(head2);

    node *merged_list = merge_sorted_ll(head1, head2);

    printf("\nMerged List:\n");
    traverse(merged_list);

    return 0;
}
```

OUTPUT:

```
List 1
4 -> 8 -> 15 -> 19 -> (NULL)

List 2
7 -> 9 -> 10 -> 16 -> (NULL)

Merged List:
4 -> 7 -> 8 -> 9 -> 10 -> 15 -> 16 -> 19 -> (NULL)
iiit@iiit-OptiPlex-3090:~/Desktop/New Folder/23BCS123/
```