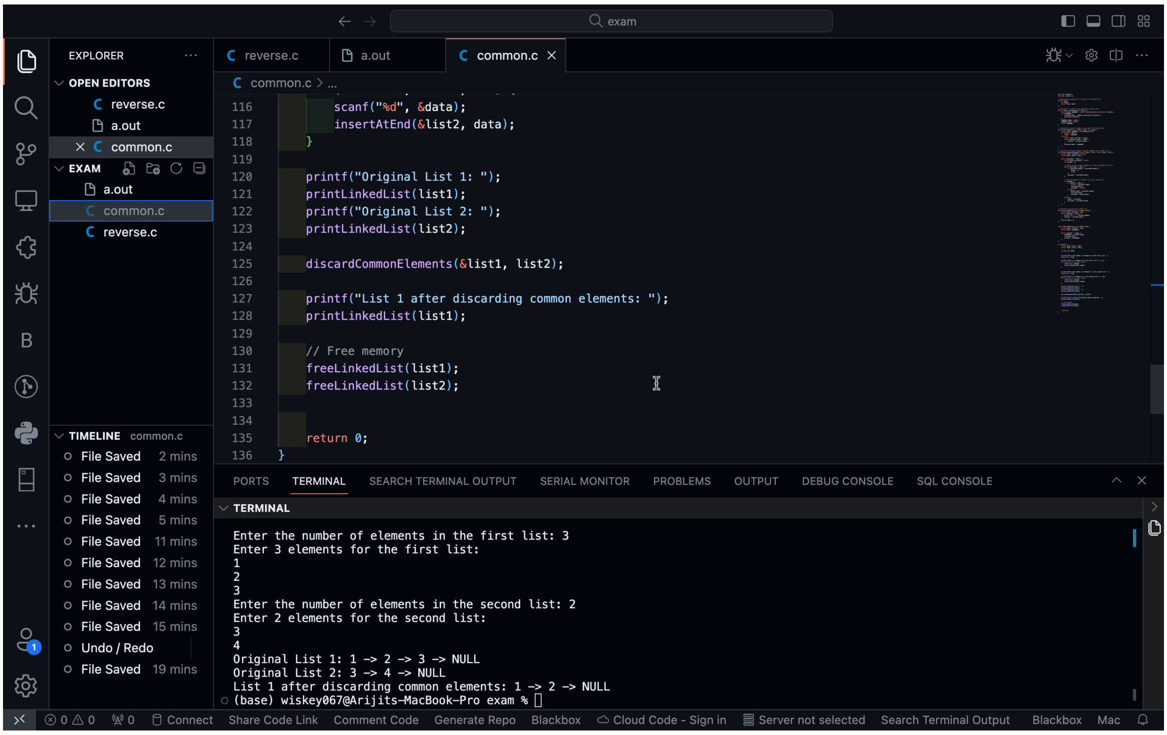
1)WAP to discard the Common Elements present between two SLL.

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

// Define the structure for a node in the linked list
struct Node {
    int data;
    struct Node* next;
};

// Function to create a new node with a given value
struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (!newNode) {
        fprintf(stderr, "Memory allocation failed\n");
        exit(EXIT_FAILURE);
    }
    newNode->data = value;
    newNode->next = NULL;
    return newNode;
}

// Function to insert a node at the end of the linked list
void insertAtEnd(struct Node** head, int value) {
    struct Node* newNode = createNode(value);
    if (*head == NULL) {
        *head = newNode;
    } else {
        struct Node* current = *head;
        while (current->next != NULL) {
            current = current->next;
        }
        current->next = newNode;
    }
}

// Function to discard common elements between two linked lists
void discardCommonElements(struct Node** list1, struct Node* list2) {
    struct Node* current1 = *list1;
    struct Node* prev1 = NULL;

    while (current1 != NULL) {
        struct Node* current2 = list2;
        int found = 0;

        // Check if the current element in list1 is present in list2
        while (current2 != NULL) {
            if (current1->data == current2->data) {
                found = 1;
                break;
            }
            current2 = current2->next;
        }

        // If the element is present in list2, discard it
        if (found) {
            if (prev1 == NULL) {
                *list1 = current1->next;
                free(current1);
                current1 = *list1;
            } else {
                prev1->next = current1->next;
                free(current1);
                current1 = prev1->next;
            }
        } else {
            prev1 = current1;
            current1 = current1->next;
        }
    }
}

// Function to print the linked list
void printLinkedList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}


void freeLinkedList(struct Node* head) {
    struct Node* current = head;
    struct Node* nextNode;

    while (current != NULL) {
        nextNode = current->next;
        free(current);
        current = nextNode;
    }
}

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

    int n1, n2, data;


    printf("Enter the number of elements in the first list: ");
    scanf("%d", &n1);

    printf("Enter %d elements for the first list:\n", n1);
    for (int i = 0; i < n1; ++i) {
        scanf("%d", &data);
        insertAtEnd(&list1, data);
    }


    printf("Enter the number of elements in the second list: ");
    scanf("%d", &n2);

    printf("Enter %d elements for the second list:\n", n2);
    for (int i = 0; i < n2; ++i) {
        scanf("%d", &data);
        insertAtEnd(&list2, data);
    }

    printf("Original List 1: ");
    printLinkedList(list1);
    printf("Original List 2: ");
    printLinkedList(list2);

    discardCommonElements(&list1, list2);

    printf("List 1 after discarding common elements: ");
    printLinkedList(list1);

    // Free memory
    freeLinkedList(list1);
    freeLinkedList(list2);


    return 0;
}
```

Output of 1st question.



```
        scanf("%d", &data);
        insertAtEnd(&list2, data);
    }

    printf("Original List 1: ");
    printLinkedList(list1);
    printf("Original List 2: ");
    printLinkedList(list2);

    discardCommonElements(&list1, list2);

    printf("List 1 after discarding common elements: ");
    printLinkedList(list1);

    // Free memory
    freeLinkedList(list1);
    freeLinkedList(list2);


    return 0;
}
```

```
Enter the number of elements in the first list: 3
Enter 3 elements for the first list:
1
2
3
Enter the number of elements in the second list: 2
Enter 2 elements for the second list:
3
4
Original List 1: 1 -> 2 -> 3 -> NULL
Original List 2: 3 -> 4 -> NULL
List 1 after discarding common elements: 1 -> 2 -> NULL
(base) wiskey067@Arijits-MacBook-Pro exam %
```

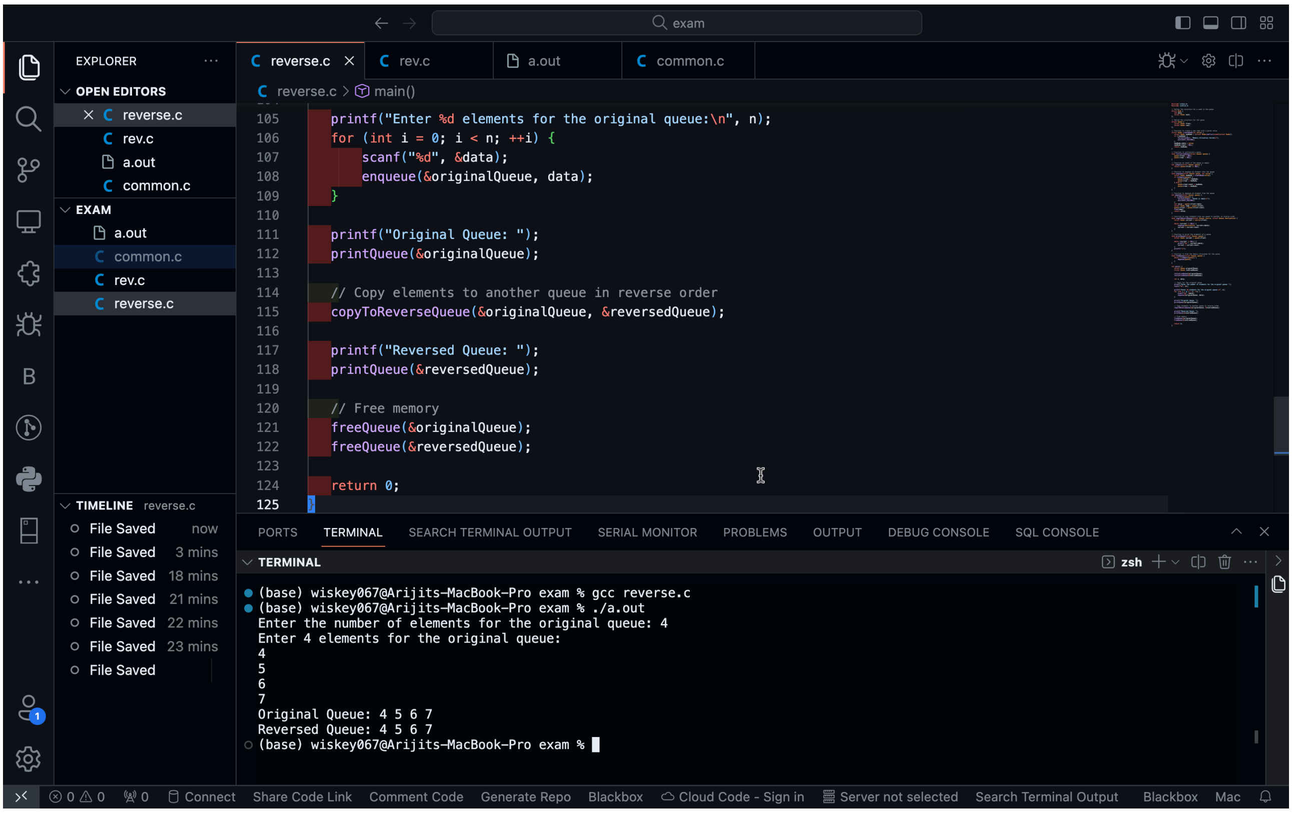
2)WAP to copy the elements of one queue to another another in reverse order.

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

// Define the structure for a node in the queue
```

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

// Define the structure for the queue
struct Queue {
    struct Node* front;
    struct Node* rear;
};

// Function to create a new node with a given value
struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (!newNode) {
        fprintf(stderr, "Memory allocation failed\n");
        exit(EXIT_FAILURE);
    }
    newNode->data = value;
    newNode->next = NULL;
    return newNode;
}

// Function to initialize a queue
void initializeQueue(struct Queue* queue) {
    queue->front = NULL;
    queue->rear = NULL;
}

// Function to check if the queue is empty
int isEmpty(struct Queue* queue) {
    return queue->front == NULL;
}

// Function to enqueue an element into the queue
void enqueue(struct Queue* queue, int value) {
    struct Node* newNode = createNode(value);
    if (isEmpty(queue)) {
        queue->front = newNode;
        queue->rear = newNode;
    } else {
        queue->rear->next = newNode;
        queue->rear = newNode;
    }
}
```

```
// Function to dequeue an element from the queue
int dequeue(struct Queue* queue) {
    if (isEmpty(queue)) {
        fprintf(stderr, "Queue is empty\n");
        exit(EXIT_FAILURE);
    }
    int value = queue->front->data;
    struct Node* temp = queue->front;
    queue->front = queue->front->next;
    free(temp);
    return value;
}

// Function to copy elements from one queue to another in reverse order
void copyToReverseQueue(struct Queue* source, struct Queue* destination)
{
    struct Node* current = source->front;

    while (current != NULL) {
        enqueue(destination, current->data);
        current = current->next;
    }
}

// Function to print the elements of a queue
void printQueue(struct Queue* queue) {
    struct Node* current = queue->front;

    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}

// Function to free the memory allocated for the queue
void freeQueue(struct Queue* queue) {
    while (!isEmpty(queue)) {
        dequeue(queue);
    }
}

int main() {
    struct Queue originalQueue;
    struct Queue reversedQueue;

    initializeQueue(&originalQueue);
    initializeQueue(&reversedQueue);

    int n, data;

    // Input for the original queue
    printf("Enter the number of elements for the original queue: ");
    scanf("%d", &n);

    printf("Enter %d elements for the original queue:\n", n);
    for (int i = 0; i < n; ++i) {
        scanf("%d", &data);
        enqueue(&originalQueue, data);
    }

    printf("Original Queue: ");
    printQueue(&originalQueue);

    // Copy elements to another queue in reverse order
    copyToReverseQueue(&originalQueue, &reversedQueue);

    printf("Reversed Queue: ");
    printQueue(&reversedQueue);

    // Free memory
    freeQueue(&originalQueue);
    freeQueue(&reversedQueue);

    return 0;
}
```

Output for 2nd question



```
    printf("Enter %d elements for the original queue:\n", n);
    for (int i = 0; i < n; ++i) {
        scanf("%d", &data);
        enqueue(&originalQueue, data);
    }

    printf("Original Queue: ");
    printQueue(&originalQueue);

    // Copy elements to another queue in reverse order
    copyToReverseQueue(&originalQueue, &reversedQueue);

    printf("Reversed Queue: ");
    printQueue(&reversedQueue);

    // Free memory
    freeQueue(&originalQueue);
    freeQueue(&reversedQueue);

    return 0;
}
```

```
(base) wiskey067@Arijits-MacBook-Pro exam % gcc reverse.c
(base) wiskey067@Arijits-MacBook-Pro exam % ./a.out
Enter the number of elements for the original queue: 4
Enter 4 elements for the original queue:
4
5
6
7
Original Queue: 4 5 6 7
Reversed Queue: 4 5 6 7
(base) wiskey067@Arijits-MacBook-Pro exam %
```

Arijit Bhattacharjee
Sec cs 23 roll 2205452