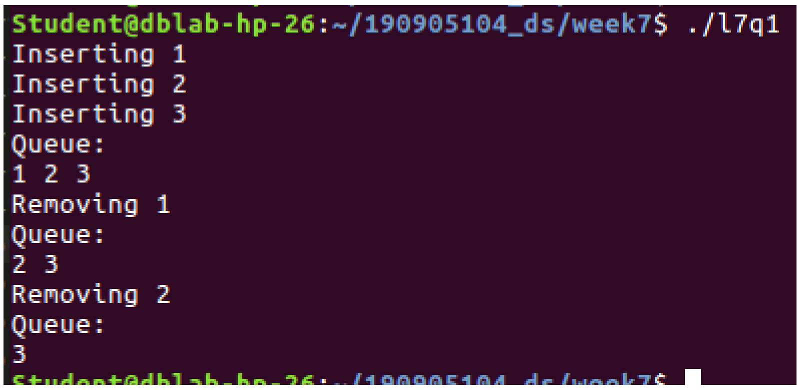
Week 7

1)

```
// Implement a queue using singly linked list without header node.

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

struct Node
{
        int data;
        struct Node *next;
}*rear = NULL, *front = NULL;

int isEmpty (){
        return rear == NULL && front == NULL;
}

struct Node * newNode (int item){
        struct Node *node = (struct Node *)malloc(sizeof(struct Node));

        if (node != NULL){
                node->data = item;
                node->next = NULL;
                return node;
        }
        else
        {
        printf ("Overflow\n");
        exit(0);
        }
}

void enqueue (int item){
        struct Node *node = newNode(item);

        if (front == NULL){
                front = node;
                rear = node;
        }
        else{
                rear->next = node;
                rear = node;
        }
}

int dequeue(){
        if (front == NULL){
                printf ("Underflow\n");
                exit (0);
        }
```

```c
	struct Node *temp = front;
	front = front->next;
	int item = temp->data;
	free (temp);
	return item;
}

void display(){
	if(isEmpty()){
		printf("Empty queue\n");
		return;
	}
	printf("Queue: \n");
	struct Node* temp = front;
	while(1){
		printf("%d ", temp->data);
		if(temp == rear) break;
		temp = temp->next;
	}
	printf("\n");
}

int peek (){
	if (front != NULL){
		return front->data;
	}
	else{
		exit (0);
	}
}


int main (){
	printf("Inserting 1\n");
	enqueue (1);
	printf("Inserting 2\n");
	enqueue (2);
	printf("Inserting 3\n");
	enqueue (3);
	display();
	printf("Removing 1\n");
	int a = dequeue();
	display();
	printf("Removing 2\n");
	int b = dequeue();
	display();

	return 0;
}
```



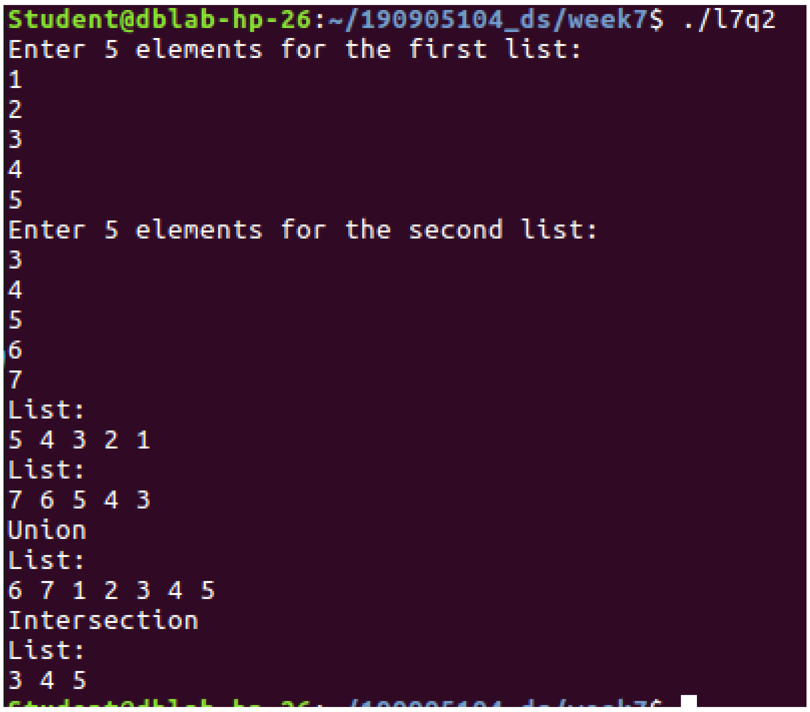
```
Student@dblab-hp-26:~/190905104_ds/week7$ ./l7q1
Inserting 1
Inserting 2
Inserting 3
Queue:
1 2 3
Removing 1
Queue:
2 3
Removing 2
Queue:
3
Student@dblab-hp-26:~/190905104_ds/week7$
```

2)

// Perform UNION and INTERSECTION set operations on singly linked lists with header node.

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

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

void display(struct Node *node){
	printf("List: \n");
	while(node != NULL){
		printf("%d ", node->data);
		node = node->next;
	}
	printf("\n");
}

int push(struct Node ** head, int data){
	if(head == NULL){
		struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
		new_node->data = data;
		new_node->next = NULL;
		*head = new_node;
	}
	struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
	new_node->data = data;
	new_node->next = (*head);
	(*head) = new_node;
}

int isPresent(struct Node *head, int data){
	struct Node *temp = head;
	while(temp != NULL){
		if(temp->data == data)
			return 1;
		temp = temp->next;
	}
	return 0;
}

struct Node * getUnion (struct Node *head1, struct Node *head2){
	struct Node *u = NULL;
	struct Node *t1 = head1, *t2 = head2;
	while (t1 != NULL){
		push (&u, t1->data);
		t1 = t1->next;
	}
	while (t2 != NULL){
		if (!isPresent (u, t2->data))
			push (&u, t2->data);
		t2 = t2->next;
	}
	return u;
}

struct Node * getIntersection (struct Node *head1, struct Node *head2){
	struct Node *i = NULL;
	struct Node *t1 = head1;
	while (t1 != NULL){
		if (isPresent (head2, t1->data))
			push (&i, t1->data);
		t1 = t1->next;
	}
	return i;
}

int main(){
	printf("Enter 5 elements for the first list: \n");
	int n;
	struct Node *h1 = NULL;
	for(int i = 0; i < 5; i++){
		scanf("%d", &n);
		push(&h1, n);
	}

	printf("Enter 5 elements for the second list: \n");
	struct Node *h2 = NULL;
	for(int i = 0; i < 5; i++){
		scanf("%d", &n);
		push(&h2, n);
	}

	display(h1);
	display(h2);
```

```
        struct Node *un = getUnion(h1, h2);
        struct Node *intersection = getIntersection(h1, h2);
        printf("Union\n");
        display(un);
        printf("Intersection\n");
        display(intersection);
        return 0;
}
```



```
Student@dblab-hp-26:~/190905104_ds/week7$ ./l7q2
Enter 5 elements for the first list:
1
2
3
4
5
Enter 5 elements for the second list:
3
4
5
6
7
List:
5 4 3 2 1
List:
7 6 5 4 3
Union
List:
6 7 1 2 3 4 5
Intersection
List:
3 4 5
```