}

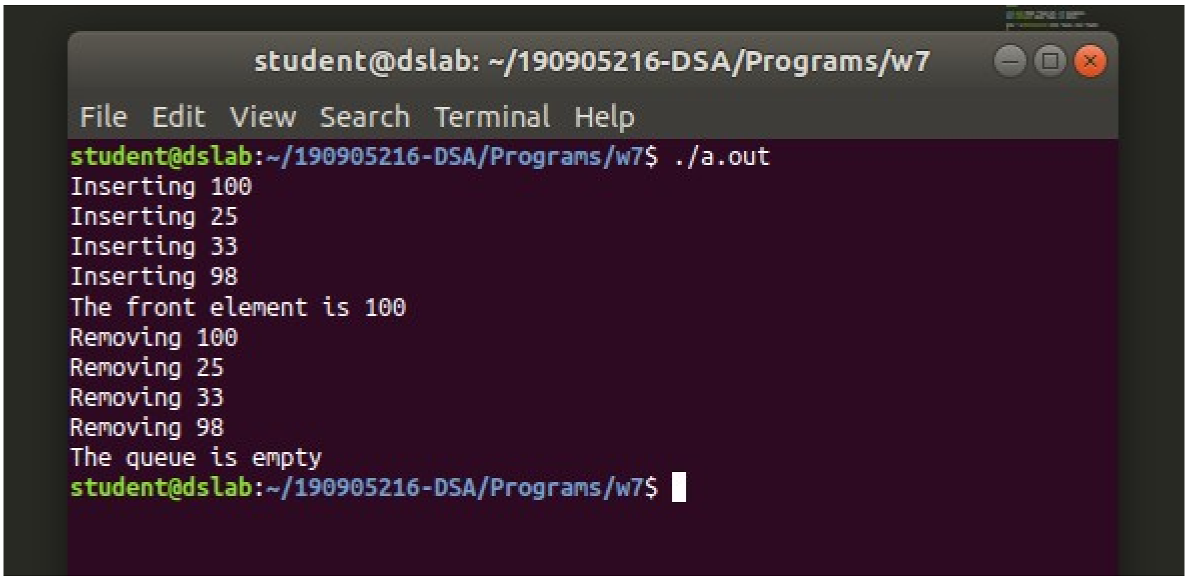
Week 7:

Q1:

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

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

Node* rear = NULL;
Node* front = NULL;

Node* newNode (int item) {

	Node* node = (Node*) malloc(sizeof(Node));

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

int dequeue() {

	if (front == NULL) {
		printf ("\nQueue Underflow");
		exit (EXIT_FAILURE);
	}

	Node* temp = front;
	printf ("Removing %d\n", temp->data);
	front = front->next;
	{
		rear = NULL;
	}

	int item = temp->data;
	free (temp);
	return item;
}
```

```
}

void enqueue (int item) {

        Node* node = newNode(item);
        printf ("Inserting %d\n", item);

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

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

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

int main () {
        enqueue (100);
        enqueue (25);
        enqueue (33);
        enqueue (98);
        printf ("The front element is %d\n", peek ());
        dequeue ();
        dequeue ();
        dequeue ();
        dequeue ();
        if (isEmpty ()){
                printf ("The queue is empty\n");
        }
        else{
                printf ("The queue is not empty\n");
        }
        return 0;
}
```

Output:


student@dslab: ~/190905216-DSA/Programs/w7
File Edit View Search Terminal Help

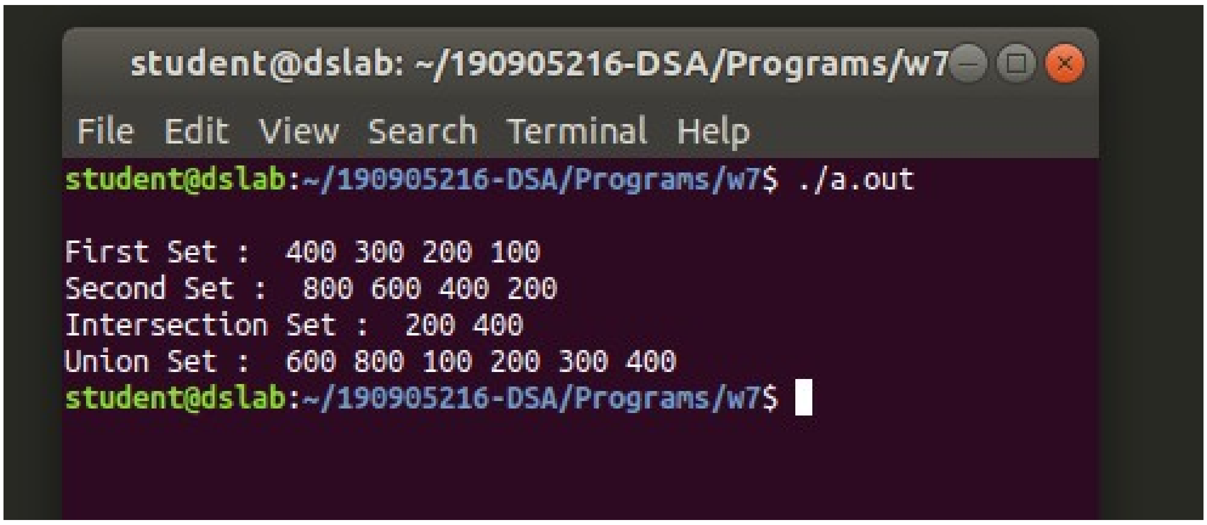
```
student@dslab:~/190905216-DSA/Programs/w7$ ./a.out
Inserting 100
Inserting 25
Inserting 33
Inserting 98
The front element is 100
Removing 100
Removing 25
Removing 33
Removing 98
The queue is empty
student@dslab:~/190905216-DSA/Programs/w7$
```

Q2:

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

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


//function signatures
void push (Node **head_ref, int new_data);
bool isPresent (Node *head, int data);

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

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

void push (Node **head_ref, int new_data)
{
        Node *new_node = (Node *) malloc (sizeof (Node));
        new_node->data = new_data;
        new_node->next = (*head_ref);
        (*head_ref) = new_node;
}
void display (Node *node)
{
        while (node != NULL)
        {
                printf ("%d ", node->data);
                node = node->next;
        }
}
bool isPresent (Node *head, int data)
{
        Node *t = head;
        while (t != NULL)
        {
                if (t->data == data)return 1;
                t = t->next;
        }
        return 0;
}
```

```
int main ()
{
	Node *head1 = NULL;
	Node *head2 = NULL;
	Node *uni = NULL;
	Node *inter = NULL;

	push (&head1, 100);
	push (&head1, 200);
	push (&head1, 300);
	push (&head1, 400);
	push (&head2, 200);
	push (&head2, 400);
	push (&head2, 600);
	push (&head2, 800);

	uni = getUnion (head1, head2);
	inter = getIntersection (head1, head2);

	printf ("\nFirst Set :  ");
	display(head1);
	printf ("\nSecond Set :  ");
	display (head2);
	printf ("\nIntersection Set :  ");
	display (inter);
	printf ("\nUnion Set :  ");
	display(uni);
	printf("\n");
	return 0;
}
```

Output:


student@dslab: ~/190905216-DSA/Programs/w7
File Edit View Search Terminal Help

```
student@dslab:~/190905216-DSA/Programs/w7$ ./a.out

First Set :  400 300 200 100
Second Set :  800 600 400 200
Intersection Set :  200 400
Union Set :  600 800 100 200 300 400
student@dslab:~/190905216-DSA/Programs/w7$
```