DSA Lab

Week 7 Assignment Submission

Swamiraju Satya Praveen Varma

200905044

Batch B1

10

Ques 1 :- Implement a queue using singly linked list without header node.

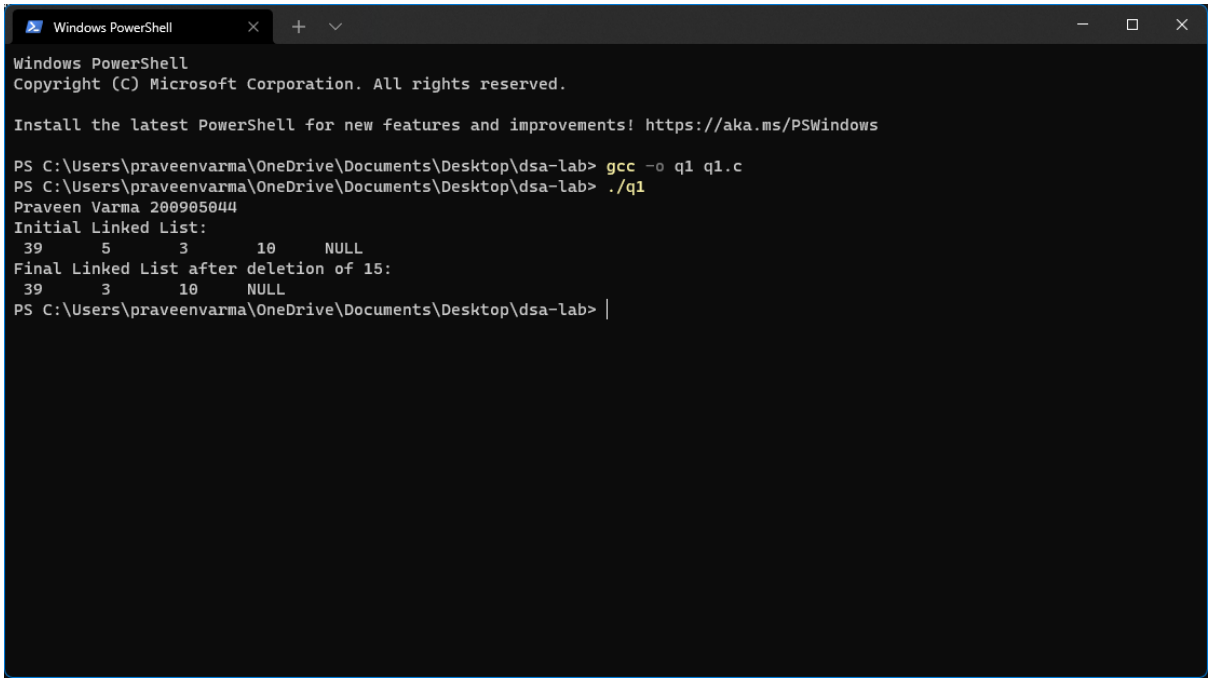
Code:

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
   int data;
   struct Node *next;
};
void deleteNodeWithoutHead(struct Node *pos)
{
   if (pos == NULL)
      return;
   else
   {
      if (pos->next == NULL)
      {
         printf("This is last node, require head, can't\n");
         return;
```

```
        }
        struct Node *temp = pos->next;
        pos->data = pos->next->data;
        pos->next = pos->next->next;
        free(temp);
    }
}
void print(struct Node *head)
{
    struct Node *temp = head;
    while (temp)
    {
        printf(" %d\t", temp->data);
        temp = temp->next;
    }
    printf("NULL");
}
void push(struct Node **head_ref, int new_data)
{
    struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}
int main()
{
```

```
    printf("Praveen Varma 200905044\n");
    struct Node *head = NULL;
    push(&head, 10);
    push(&head, 3);
    push(&head, 5);
    push(&head, 39);
    printf("Initial Linked List: \n");
    print(head);
    struct Node *del = head->next;
    deleteNodeWithoutHead(del);
    printf("\nFinal Linked List after deletion of 15:\n");
    print(head);
    return 0;
}
```

Sample input/output:



Ques 2 :- Perform UNION and INTERSECTION set operations on singly linked lists with header node.

Code:

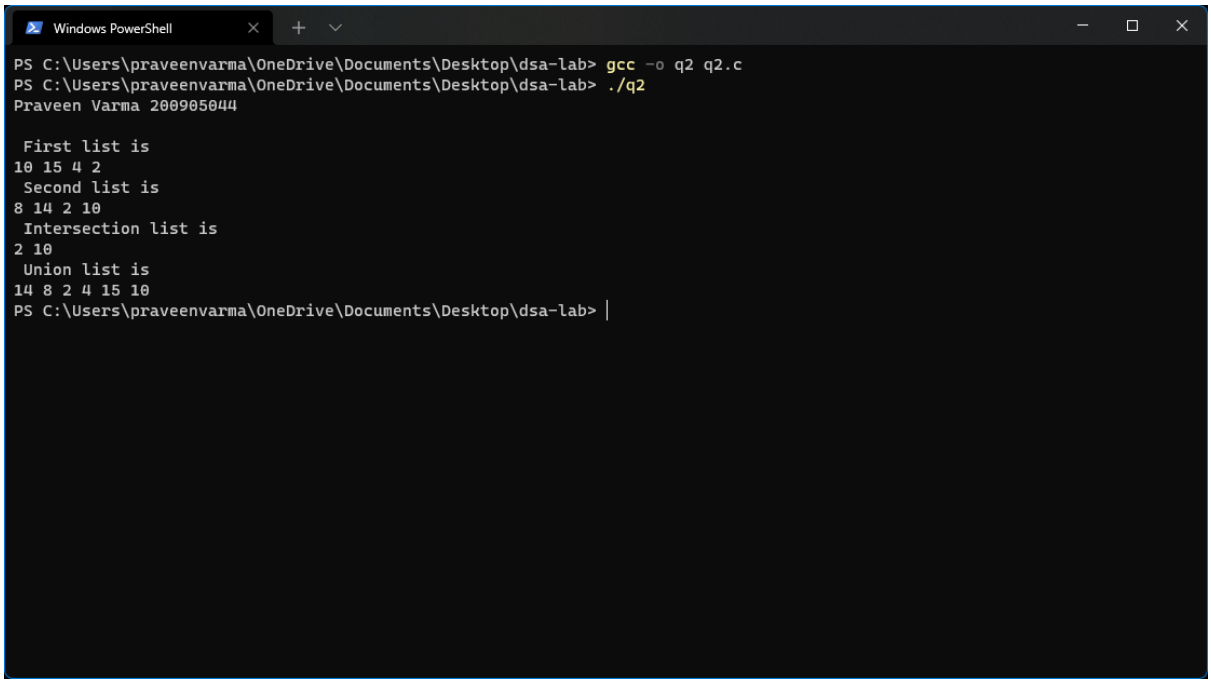
```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
struct Node
{
   int data;
   struct Node *next;
};
void push(struct Node **head_ref, int new_data);
bool isPresent(struct Node *head, int data);
struct Node *getUnion(struct Node *head1, struct Node *head2)
{
   struct Node *result = NULL;
   struct 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;
}
struct Node *getIntersection(struct Node *head1, struct Node *head2)
{
    struct Node *result = NULL;
    struct Node *t1 = head1;
    while (t1 != NULL)
    {
        if (isPresent(head2, t1->data))
            push(&result, t1->data);
        t1 = t1->next;
    }
    return result;
}
void push(struct Node **head_ref, int new_data)
```

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

```
{
    printf("Praveen Varma 200905044\n");
    struct Node *head1 = NULL;
    struct Node *head2 = NULL;
    struct Node *intersecn = NULL;
    struct Node *unin = NULL;
    push(&head1, 2);
    push(&head1, 4);
    push(&head1, 15);
    push(&head1, 10);
    push(&head2, 10);
    push(&head2, 2);
    push(&head2, 14);
    push(&head2, 8);
    intersecn = getIntersection(head1, head2);
    unin = getUnion(head1, head2);
    printf("\n First list is \n");
    printList(head1);
    printf("\n Second list is \n");
    printList(head2);
    printf("\n Intersection list is \n");
    printList(intersecn);
    printf("\n Union list is \n");
    printList(unin);
    return 0;
}
```

Sample input/output:



THANK YOU!