Data Structures and Algorithms

Lab Report

Lab06

Group Members Name & Reg #:	**Muhammad Haris Irfan** **(FA18-BCE-090)**
Class	Data Structures and Algorithms CSC211 (**BCE-3B**)
Instructor's Name	Dilshad Sabir

Pre-Lab Task

Task:1

Complete the functions 'enqueue ()', 'dequeue ()' and peek () functions.

Solution:

The code is shown below,

Dequeue Code:

```
struct element q_delete(struct node ** qfront)  /// The dequeue function
{
    /* Complete this function */
if(*qfront==NULL)
    printf("\nthe list is empty");

else
{

struct element data;
data=(*qfront)->data;
*qfront=(*qfront)->next;
return(data);
 printf("Dequeue Successful\n");
}
}
```

Enqueue Code:

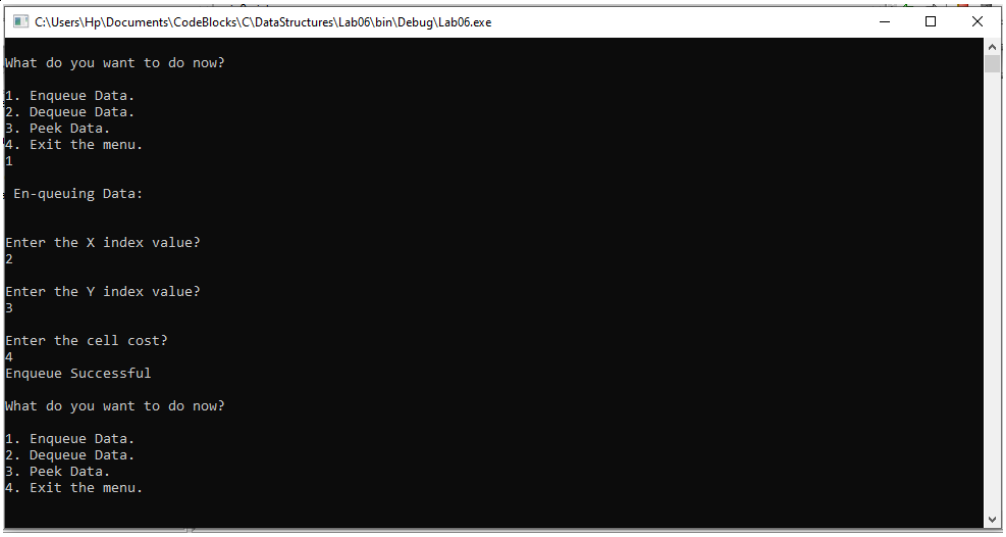
```
void q_insert(struct node ** qrear, struct node ** qfront, struct element new_data)
{
    struct node * new_node = (struct node *) malloc(sizeof(struct node));

    new_node->data = new_data;      /// I can assign one struct to another if the type is the same

    new_node->next = NULL;
    if(*qrear != NULL)              /// If the queue is currently empty
        (*qrear)->next = new_node;

    *qrear = new_node;
    if(*qfront == NULL)             /// This operation will only be perfomed when the que is empty
        *qfront = new_node;

    (*qrear)->next = NULL;
    printf("Enqueue Successful\n");
}
```

Peek Code:

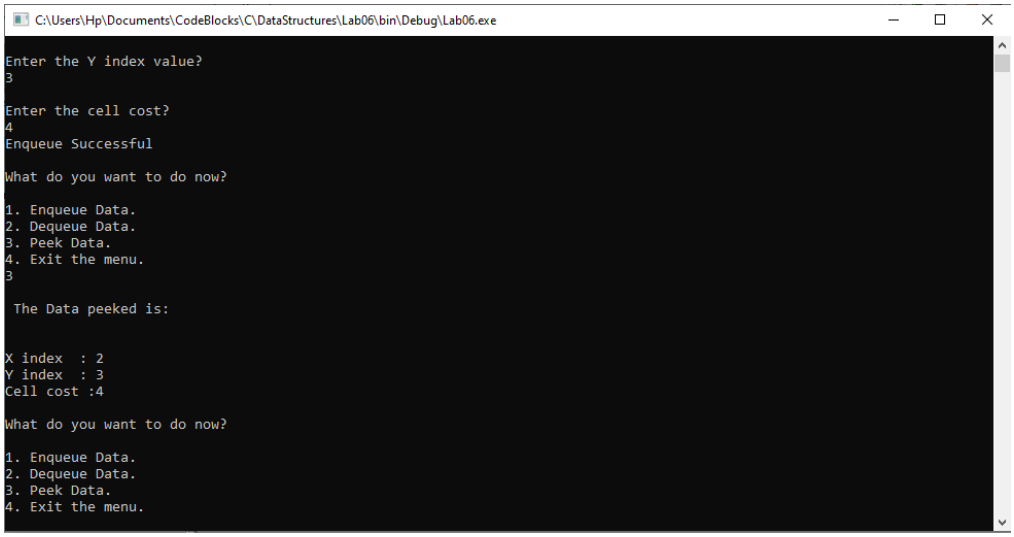
```
struct element q_peek(struct node ** qfront)
{
    if(*qfront == NULL)                     /// if a dequeue operation is sought when the queue is already empty
    {
        printf("The queue was empty. Returning garbage data! \n\n");
        struct element temp;
        return(temp);
    }
    struct element temp = (*qfront)->data;   /// I copy the data at the front node into a temporary variable

    return(temp);   /// Return the node data without deleting it from the queue
}
```

The Result of the following code is attached below:

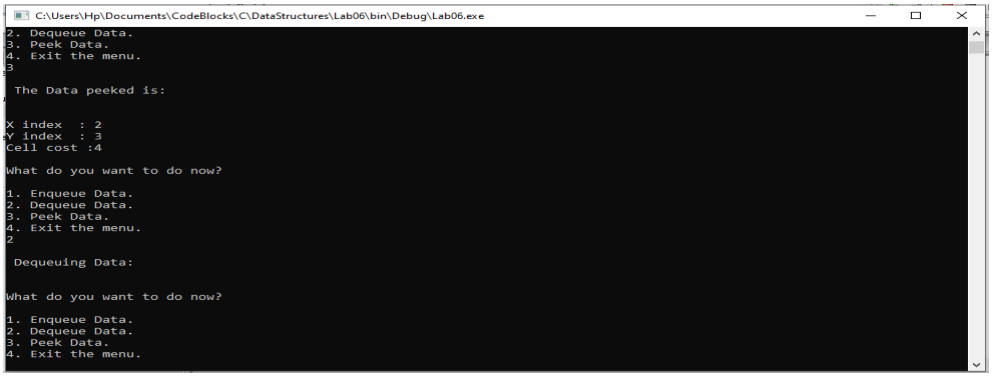
Enqueing:



Peeking:



Dequeing:



In Lab Tasks

Task:1

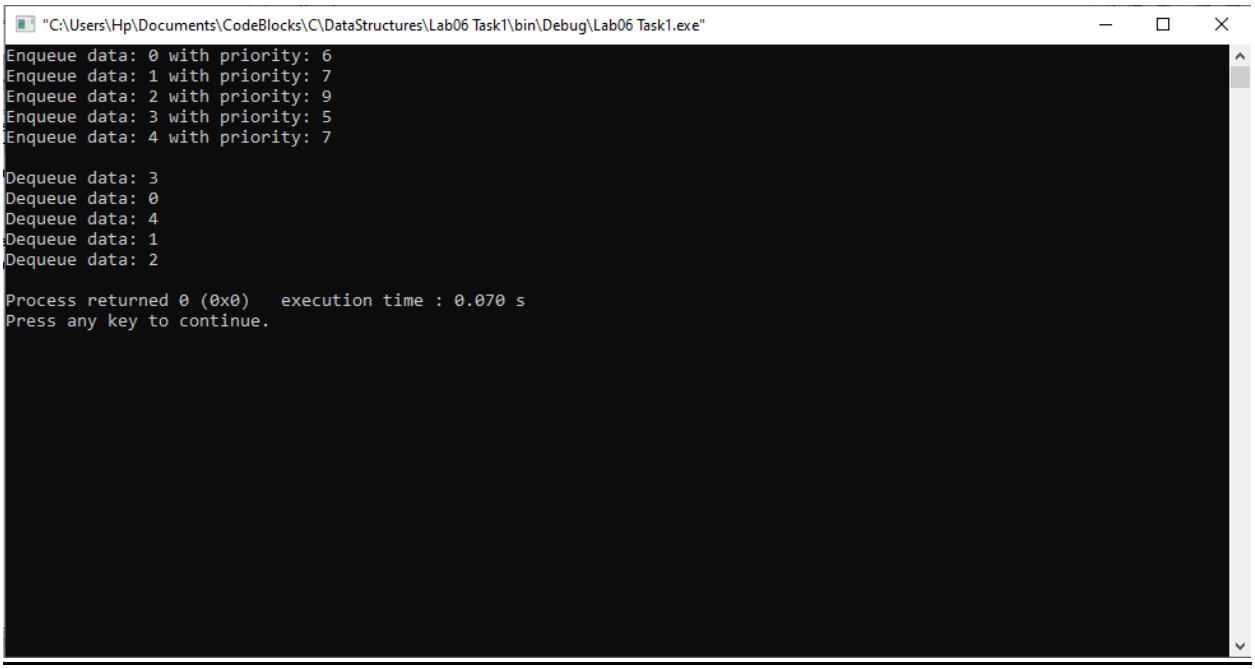
Implement a priority queue. (Enqueue Function).

Solution:

The code is shown below,

```
void pr_enqueue(struct node ** qrear, struct node ** qfront, struct element new_data, int priority)
{
    /* Complete this function*/
    struct node* begin = (*qfront);
    struct node * new_node = (struct node *) malloc(sizeof(struct node));
    new_node->data = new_data;
    new_node->node_priority = priority;
    new_node->next = NULL;
    if(*qfront == NULL)
     {
         *qfront = new_node;
         *qrear = new_node;
    (*qrear)->next = NULL;
     }
     else
     {
    struct node * temp= new_node;

     if ((*qfront)->node_priority > priority)
    {
        temp->next = *qfront;
        (*qfront) = temp;
    }
    else
    {
        while (begin->next != NULL && begin->next->node_priority < priority) {
            begin = begin->next;
        }
        temp->next = begin->next;
        begin->next = temp;
    }
     }
}
```

The Result of the following code is attached below:



Task:2

Find the Shortest Path in Graphs Using BFS and Queues. (Avoiding 1s)

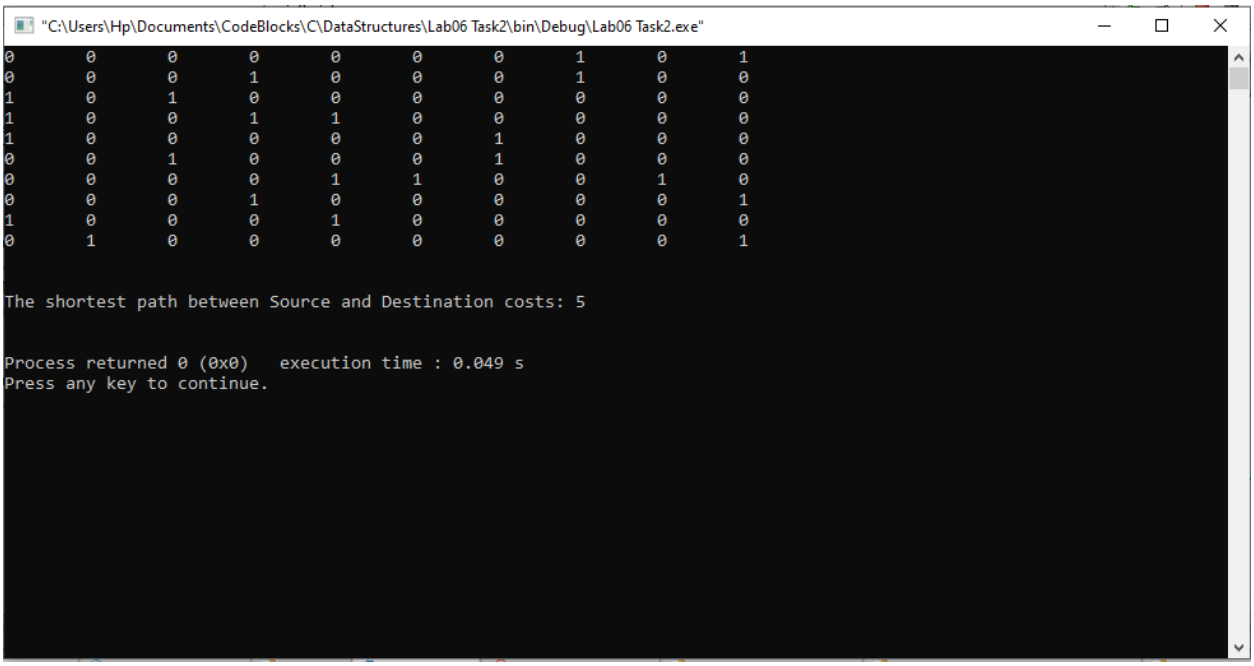
Solution

The code is shown below,

Only this condition was enough to avoid the ones and find the shortest path.

```
if(visited[src_x][src_y] ==0)
{
    visited[src_x][src_y] = 1;
}
```

The Result of the following code is attached below:



THE END