Activity No. 5.1	
Queues	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:11/09/25**
Section: CPE21S4	**Date Submitted:11/09/25**
Name(s): Crishen Luper S. Pulgado	**Instructor:**

6. Output

```cpp
#include <iostream>
#include <queue>
#include <string>


void display(std::queue<std::string> q) {
    std::queue<std::string> temp = q;
    while (!temp.empty()) {
        std::cout << temp.front() << " ";
        temp.pop();
    }
    std::cout << std::endl;
}

int main() {
    std::string students[] = {"Nikko", "David", "Jeus", "Heidee", "Akira"};
    int n = sizeof(students) / sizeof(students[0]);
    std::queue<std::string> studentQueue;
    for (int i = 0; i < n; i++) {
        studentQueue.push(students[i]);
    }
    std::cout << "Students in queue: ";
    display(studentQueue);

    std::cout << "Front of queue: " << studentQueue.front() << std::endl;
    std::cout << "Back of queue: " << studentQueue.back() << std::endl;

    std::cout << "Removing: " << studentQueue.front() << std::endl;
    studentQueue.pop();

    std::cout << "Queue after one removal: ";
    display(studentQueue);

    return 0;
}

```

```
Z:\Crishen\HOA_main.exe
Students in queue: Nikko David Jeus Heidee Akira
Front of queue: Nikko
Back of queue: Akira
Removing: Nikko
Queue after one removal: David Jeus Heidee Akira

--------------------------------
Process exited after 0.08417 seconds with return value 0
Press any key to continue . . .
```

Similarly to the example code, I put a display function wherein it takes the queue by value so that the original queue in the main cannot be modified when it is pop. It is also added a copy of queue named temp so that it removes the element easily. The while loop runs when it is not empty. The array has 5 elements and the integer n calculates the number of elements in the array. The numerator as the total memory size of an array and the denominator is the memory size of one element which gives it 5. In the code, I call the display to display the elements in the array and print the front and back of the queue. Once I popped the queue, it popped Nikko and the array is left with David, Jeus, Heidee, and Akira. ILO B.1

```
public:
    Queue() : frontPtr(NULL), backPtr(NULL) {}

    void display() {
        if (frontPtr == NULL) {
            std::cout << "Queue is empty.\n";
            return;
        }
        Node* temp = frontPtr;
        while (temp != NULL) {
            std::cout << temp->data << " ";
            temp = temp->next;
        }
        std::cout << "\n";
    }

    void insertNonEmpty(int value) {
        Node* newPtr = new Node(value);
        newPtr->next = NULL;
        backPtr->next = newPtr;
        backPtr = newPtr;
        std::cout << "Inserted " << value << " into a non-empty queue.\n";
        display();
    }

    void insertEmpty(int value) {
        Node* newPtr = new Node(value);
        frontPtr = newPtr;
        backPtr = newPtr;
        std::cout << "Inserted " << value << " into an empty queue.\n";
        display();
    }

    void deleteMultiple() {
```

Compile Log | Debug | Find Results | Close

```
Select Z:\Crishen\QUEUELINK.exe
Inserted 2 into an empty queue.
2
Inserted 4 into a non-empty queue.
2 4
Inserted 1 into a non-empty queue.
2 4 1
Inserted 7 into a non-empty queue.
2 4 1 7
Inserted 3 into a non-empty queue.
2 4 1 7 3
Deleted 2 from queue with multiple items.
4 1 7 3
Deleted 4 from queue with multiple items.
1 7 3
Deleted 1 from queue with multiple items.
7 3
Deleted 7 from queue with multiple items.
3
Deleted 3 from queue with one item.
Queue is empty.

--------------------------------
Process exited after 0.08153 seconds with return value 0
Press any key to continue . . .
```

```
void deleteMultiple() {
    if (frontPtr == NULL || frontPtr->next == NULL) {
        std::cout << "Operation invalid: not enough items.\n";
        return;
    }
    Node* tempPtr = frontPtr;
    frontPtr = frontPtr->next;
    tempPtr->next = NULL;
    std::cout << "Deleted " << tempPtr->data << " from queue with multiple items.\n
    delete tempPtr;
    display();
}

void deleteSingle() {
    if (frontPtr == NULL || frontPtr->next != NULL) {
        std::cout << "Operation invalid: queue does not have exactly one item.\n";
        return;
    }
    Node* tempPtr = frontPtr;
    frontPtr = NULL;
    backPtr = NULL;
    std::cout << "Deleted " << tempPtr->data << " from queue with one item.\n";
    delete tempPtr;
    display();
}
```

```
int main() {
    Queue q;

    q.insertEmpty(2);

    q.insertNonEmpty(4);
    q.insertNonEmpty(1);
    q.insertNonEmpty(7);
    q.insertNonEmpty(3);

    q.deleteMultiple();

    q.deleteMultiple();
    q.deleteMultiple();
    q.deleteMultiple();

    q.deleteSingle();

    return 0;
}
```

ILO B.2

Supplementary:

7. Supplementary Activity

8. Conclusion

In conclusion, this activity has taught us a linear data structure called queue that operates the principle of the First-in, first-out. Queues can be implemented using the STL, linked list, and array based circular queues. The STL has built in operations like enqueue(push), dequeue(pop), front, back, size, and empty. For the linked list procedure, it uses dynamic memory allocation to create nodes, making insertion and deletion. For the array circular, it relies on data members such as q_array, q_capacity, q_size, queue_front, q_back. I think I did well in understanding the concepts of queue but the implementation in arrays and linked list is what I lack in this activity.

9. Assessment Rubric