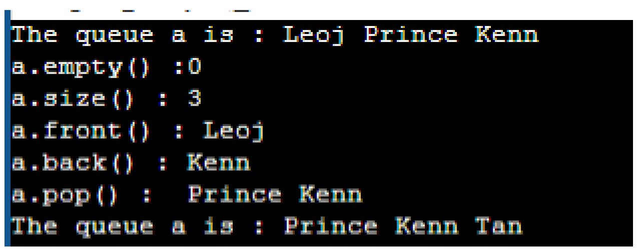
Activity No. 5	
Queues	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: 07 / 10 / 24**
Section:CpE21S4	**Date Submitted: 07 / 10 / 24**
Name(s): Leoj Jeam B. Tandayu	**Instructor: Ms. Ma. Rizette Sayo**
6. Output	

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

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

int main() {
std::queue<std::string> a;
a.push("Leoj");
a.push("Prince");
a.push("Kenn");
std::cout << "The queue a is :";
display(a);
std::cout << "a.empty() :" << a.empty() << "\n";
std::cout << "a.size() : " << a.size() << "\n";
std::cout << "a.front() : " << a.front() << "\n";
std::cout << "a.back() : " << a.back() << "\n";
std::cout << "a.pop() : ";
a.pop();
display(a);
a.push("Tan");
std::cout << "The queue a is :";
display(a);
return 0;
}
```



```
The queue a is : Leoj Prince Kenn
a.empty() :0
a.size() : 3
a.front() : Leoj
a.back() : Kenn
a.pop() :  Prince Kenn
The queue a is : Prince Kenn Tan
```

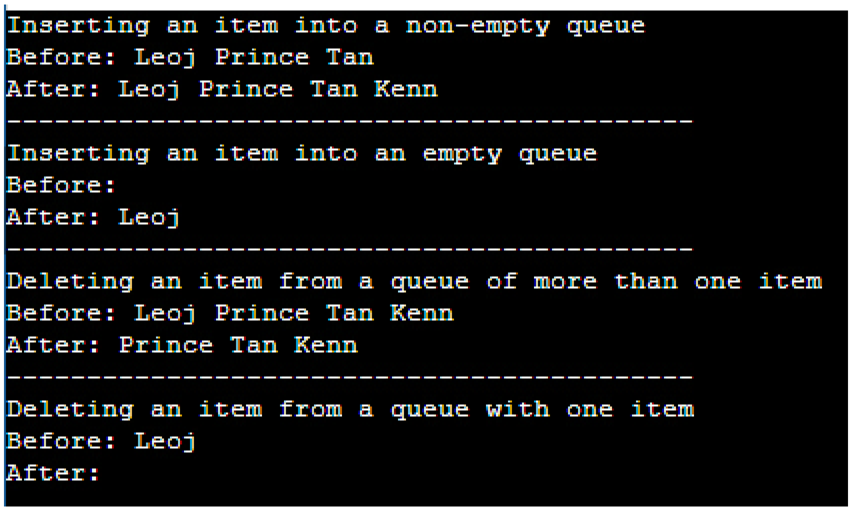
To make the modification work, I first converted the int to string, then proceeded to change the numbers into names.

Table 5-1. Queues using C++ STL

```
#include <iostream>
#include <string>

using namespace std;

// Node structure for the linked list
struct Node {
    string data;
    Node* next;
};

// Queue class using a linked list
class Queue {
private:
    Node* front;
    Node* rear;
    int size;

public:
// Constructor to initialize the queue
Queue() : front(nullptr), rear(nullptr), size(0) {}

// Destructor to free the memory
~Queue() {
while (front != nullptr) {
Node* temp = front;
front = front->next;
delete temp;
}
}

// Function to check if the queue is empty
bool isEmpty() {
return (size == 0);
}

// Function to get the front element of the queue
string getFront() {
if (isEmpty()) {
return "";
}
return front->data;
}

// Function to get the rear element of the queue
```

```
string getRear() {
if (isEmpty()) {
return "";
}
return rear->data;
}

// Function to insert an item into the queue
void enqueue(const string& item) {
Node* newNode = new Node();
newNode->data = item;
newNode->next = nullptr;

if (isEmpty()) {
front = rear = newNode;
} else {
rear->next = newNode;
rear = newNode;
}
size++;
}

// Function to delete an item from the queue
void dequeue() {
if (isEmpty()) {
return;
}
Node* temp = front;
front = front->next;
delete temp;
size--;

if (isEmpty()) {
rear = nullptr;
}
}

// Function to display the queue
void display() {
Node* temp = front;
while (temp != nullptr) {
cout << temp->data << " ";
temp = temp->next;
}
cout << "\n";
}
};

int main() {
    Queue q;
```

```
// Inserting items in a non empty queue
cout << "Inserting an item into a non-empty queue" << endl;
q.enqueue("Leoj");
q.enqueue("Prince");
q.enqueue("Tan");
cout << "Before: ";
q.display();
q.enqueue("Kenn");
cout << "After: ";
q.display();
cout<<"-------------------------------------------"<<endl;
// Inserting items in a empty queue
cout << "Inserting an item into an empty queue" << endl;
Queue q2;
cout << "Before: ";
q2.display();
q2.enqueue("Leoj");
cout << "After: ";
q2.display();
cout<<"-------------------------------------------"<<endl;
// Deleting from a queue with more than 1 item
cout << "Deleting an item from a queue of more than one item" << endl;
Queue q3;
q3.enqueue("Leoj");
q3.enqueue("Prince");
q3.enqueue("Tan");
q3.enqueue("Kenn");
cout << "Before: ";
q3.display();
q3.dequeue();
cout << "After: ";
q3.display();
cout<<"-------------------------------------------"<<endl;
// Deleting from a queue with 1 item
cout << "Deleting an item from a queue with one item" << endl;
Queue q4;
q4.enqueue("Leoj");
cout << "Before: ";
q4.display();
q4.dequeue();
cout << "After: ";
q4.display();

return 0;
}
```

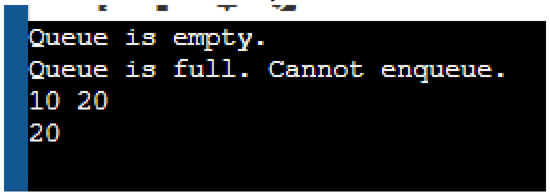


```
Inserting an item into a non-empty queue
Before: Leoj Prince Tan
After: Leoj Prince Tan Kenn
-------------------------------------------
Inserting an item into an empty queue
Before:
After: Leoj
-------------------------------------------
Deleting an item from a queue of more than one item
Before: Leoj Prince Tan Kenn
After: Prince Tan Kenn
-------------------------------------------
Deleting an item from a queue with one item
Before: Leoj
After:
```

Table 5-2. Queues using Linked List Implementation

```
#include <iostream>
#include <string>
using namespace std;

const int MAX_SIZE = 2;

int queueArray[MAX_SIZE];
int front = 0;
int rear = 0;
int queueSize = 0;

void enqueue(int item) {
if (queueSize == MAX_SIZE) {
cout << "Queue is full. Cannot enqueue." << endl;
return;
}
queueArray[rear] = item;
rear = (rear + 1);
queueSize++;
}

void dequeue() {
if (queueSize == 0) {
cout << "Queue is empty. Cannot dequeue." << endl;
return;
}
front = (front + 1);
queueSize--;
}

void display() {
```

```
if (queueSize == 0) {
cout << "Queue is empty." << endl;
return;
}
int temp = front;
for (int i = 0; i < queueSize; i++) {
cout << queueArray[temp] << " ";
temp = (temp + 1);
}
cout << endl;
}

int main() {
display();
enqueue(10);
enqueue(20);
enqueue(30);
display();
dequeue();
display();
return 0;
}
```

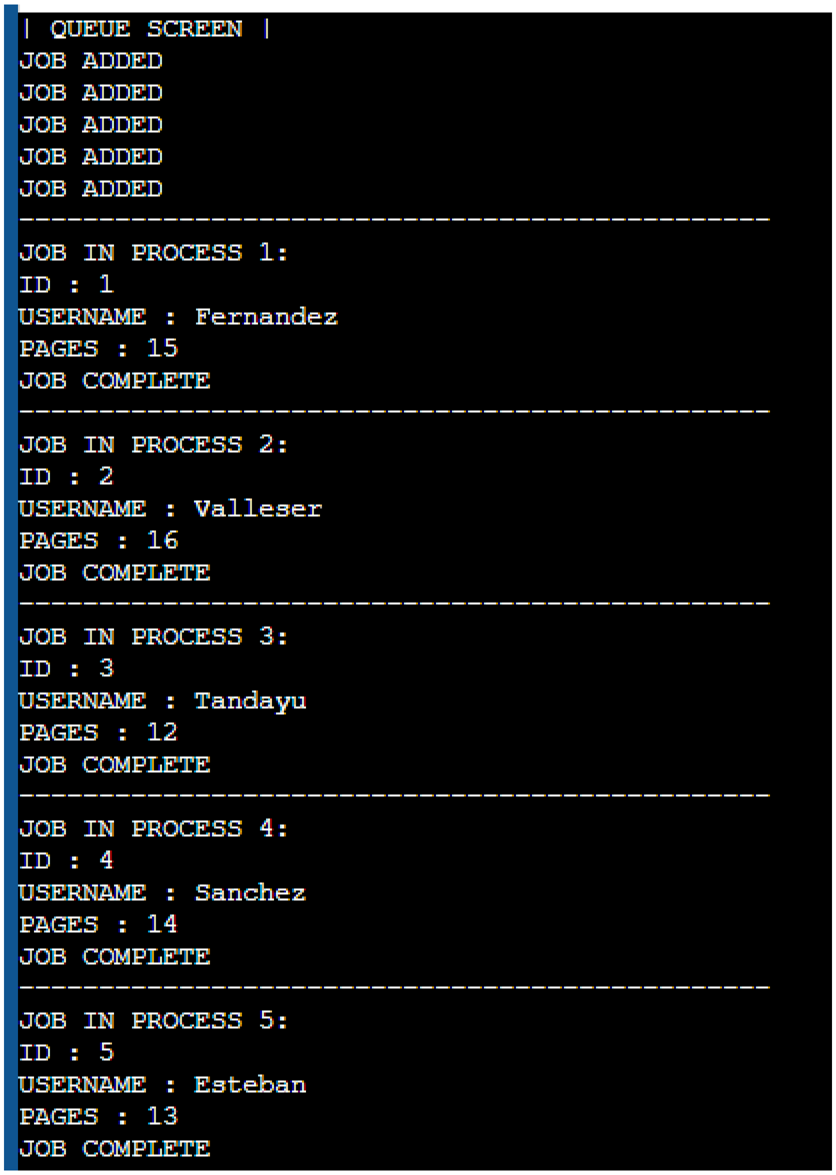


```
Queue is empty.
Queue is full. Cannot enqueue.
10 20
20
```

Table 5-3. Queues using Array Implementation

7. Supplementary Activity

```
#include <iostream>
#include <string>

using namespace std;

class Job // create a class called Job
{
private:
    int Id;
    string User;
    int num_page;

public:
    Job(int id, string userName, int pages)
        {
        this->Id = id;
        this->User  = userName;
        this->num_page = pages;
        }
    void display_job()
    {
        cout << "ID : " << Id << endl;
        cout << "USERNAME : " << User << endl;
        cout << "PAGES : " << num_page << endl;
    }
};

class Printer // create a class called Printer
{
private:
    Job** jobs;
    int capacity;
    int in_line;

public:
    Printer(int capacity)
    {
        this->capacity = capacity;
        this->in_line = 0;
        this->jobs = new Job*[capacity];
    }

    ~Printer() {
        for (int i = 0; i < in_line; i++) {
            delete jobs[i];
        }
        delete[] jobs;
    }

    void add_job(Job* job)
    {

        if (in_line < capacity)
        {
            jobs[in_line] = job;
            in_line++;
            cout << "JOB ADDED" << endl;
        }
        else
        {
            cout << "LIMIT EXCEEDED" << endl;
        }
    }

    void process_job()
    {
        for (int i = 0 ; i < in_line ; i++)
        {
            cout << "----------------------------------------------"<<endl;
            cout << "JOB IN PROCESS " << i + 1 << ":" << endl;
            jobs[i]->display_job();
            cout << "JOB COMPLETE" << endl;

        }
    }
};

int main()
{

    cout<< "| QUEUE SCREEN | " <<endl;
    Printer printer(5);

    Job* job1 = new Job(001, "Fernandez", 15);
    Job* job2 = new Job(002, "Valleser", 16);
    Job* job3 = new Job(003, "Tandayu", 12);
    Job* job4 = new Job(004, "Sanchez", 14);
    Job* job5 = new Job(005, "Esteban", 13);

    printer.add_job(job1);
    printer.add_job(job2);
    printer.add_job(job3);
    printer.add_job(job4);
    printer.add_job(job5);

    printer.process_job();

    return 0;
}
```



```
| QUEUE SCREEN |
JOB ADDED
JOB ADDED
JOB ADDED
JOB ADDED
JOB ADDED
-----------------------------------------------
JOB IN PROCESS 1:
ID : 1
USERNAME : Fernandez
PAGES : 15
JOB COMPLETE
-----------------------------------------------
JOB IN PROCESS 2:
ID : 2
USERNAME : Valleser
PAGES : 16
JOB COMPLETE
-----------------------------------------------
JOB IN PROCESS 3:
ID : 3
USERNAME : Tandayu
PAGES : 12
JOB COMPLETE
-----------------------------------------------
JOB IN PROCESS 4:
ID : 4
USERNAME : Sanchez
PAGES : 14
JOB COMPLETE
-----------------------------------------------
JOB IN PROCESS 5:
ID : 5
USERNAME : Esteban
PAGES : 13
JOB COMPLETE
```

8. Conclusion

In conclusion, the activity has taught me how queues are implemented using c++. It showed me how a variable can be enqueued and dequeued from either list or array. The supplementary activity is quite challenging but also taught me more how the queue works as it gives me more visualization about it. I would say that I did pretty well on the activity although I create errors here and there, I manage to find and fix the problem in the syntax.

9. Assessment Rubric