Activity No. 5	
QUEUE	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** 10/07/2024
Section: CPE21S4	**Date Submitted:** 10/07/2024
Name(s): Santos, Ma. Kassandra Nicole	**Instructor:** Ma'am Rizette Sayo

6. Output

CODE AND OUTPUT	OBSERVATION

C/C++

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

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

int main() {
    queue<string> a; //instead of
integer, I replaced it with string
    a.push("Anna");
    a.push("Emma");
    a.push("Lauren");

    cout << "The queue of names 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";

    cout << "After Popping the first
name: ";
    a.pop();
    display(a);

    a.push("Danica");
```

This code shows how a queueing would normally look like, as we know queuing follows the first in and last out, so in this code we see that the first element, which is "Anna" was popped from the list and then comes in Danica, from there we see that the first name that comes in now is Emma and Danica comes in the list being at the last

```
    std::cout << "The queue a after
pushing 'Danica' is: ";
    display(a);

    return 0;
}
```

```
The queue of names is: Anna Emma Lauren
a.empty() :0
a.size() : 3
a.front() : Anna
a.back() : Lauren
After Popping the first name: Emma Lauren
The queue a after pushing 'Danica' is: Emma Lauren Danica
```

TABLE 5-1. Queues using C++ STL

CODE AND OUTPUT	OBSERVATION

C/C++

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

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

struct Node {
    string data;
    Node* next;

    Node(string value) : data(value),
next(nullptr) {}
};

class Queue {
    private:
        Node* front;
        Node* back;
    public:
    Queue() : front(nullptr),
back(nullptr) {}
```

using a linked list, we managed to input a new name but in a random place while still maintaining the queueing principles by using a node and using dequeuing and queueing

```
    ~Queue() {
        while (!isEmpty()) {
            dequeue();
        }
    }

    bool isEmpty() {
        return front == nullptr;
    }

    void enqueue(string value) {
        Node* newNode = new Node(value);
        if (back) {
            back->next = newNode;
        }
        back = newNode;
        if (!front) {
            front = newNode; // Set
front if the queue was empty
        }
    }

    string dequeue() {
        if (isEmpty()) {
            throw runtime_error("Queue
is empty");
        }
        Node* temp = front;
        string value = front->data;
        front = front->next;
        if (!front) {
            back = nullptr; // Reset
back if the queue is empty now
        }
        delete temp;
        return value;
    }

    string peek() {
        if (isEmpty()) {
            throw runtime_error("Queue
is empty");
        }
        return front->data;
    }

    void display() {
        for (Node* current = front;
current; current = current->next) {
            cout << current->data << "
";
        }
        cout << endl;
    }
```

```
};

int main() {
    queue<string> a; //instead of
integer, I replaced it with string
    a.push("Anna");
    a.push("Emma");
    a.push("Lauren");

    cout << "The queue of names 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";

    a.push ("Shalisa");
    cout << "After adding 'Olivia': ";
    display(a);

    cout << "After Popping the first
name: ";
    a.pop();
    display(a);

    a.push("Danica");
    std::cout << "The queue a after
pushing 'Danica' is: ";
    display(a);

    return 0;

 }
```

```
The queue of names is: Anna Emma Lauren
a.empty() :0
a.size() : 3
a.front() : Anna
a.back() : Lauren
After adding 'Olivia': Anna Emma Lauren Shalisa
After Popping the first name: Emma Lauren Shalisa
The queue a after pushing 'Danica' is: Emma Lauren Shalisa Danica
```

x

TABLE 5-2. Queues using Linked List Implementation

CODE AND OUTPUT	OBSERVATION

C/C++

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

void display(const queue<string>& q) {
    queue<string> temp = q; // Make a
copy of the queue
    while (!temp.empty()) {
        cout << temp.front() << " ";
        temp.pop();
    }
    cout << endl;
}

int main() {
    queue<string> a;
    a.push("Anna");
    a.push("Emma");
    a.push("Lauren");

    cout << "The queue of names is: ";
    display(a);

    cout << "a.empty() : " << a.empty()
<< "\n";
    cout << "a.size() : " << a.size() <<
"\n";
    cout << "a.front() : " << a.front()
<< "\n";
    cout << "a.back() : " << a.back() <<
"\n";

    cout << "After Popping the first
name: ";
    a.pop();
    display(a);

    a.push("Danica");
    cout << "The queue a after pushing
'Danica' is: ";
    display(a);

    return 0;
}
```

```
The queue of names is: Anna Emma Lauren
a.empty() : 0
a.size() : 3
a.front() : Anna
a.back() : Lauren
After Popping the first name: Emma Lauren
The queue a after pushing 'Danica' is: Emma Lauren Danica
```

Using array, all I did was simply changing the first part of the code which is this

C/C++

```
void display(const queue<string>& q) {
    queue<string> temp = q; // Make a
copy of the queue
    while (!temp.empty()) {
        cout << temp.front() << " ";
        temp.pop();
    }
    cout << endl;
}



//this code used to diplay the array,
the while (!temp.empty()) loop continues
as long as the temp is not empty and it
retrieves the element from the from and
removies an element from the queues
```

TABLE 5-3. Queues using Array Implementation

7. Supplementary Activity

C/C++

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

struct Job {
    int id;
    string user;
    int pages;
    Job* next;

    Job(int jobId, string userName, int numPages) : id(jobId), user(userName),
pages(numPages), next(nullptr) {}
};

class Printer {
private:
    Job* front;
    Job* back;

public:
    Printer() : front(nullptr), back(nullptr) {}

    bool isEmpty() {
        return front == nullptr;
    }

    void enqueue(int id, string user, int pages) {
        Job* newJob = new Job(id, user, pages);
        if (back) {
            back->next = newJob;
        }
        back = newJob;
        if (!front) {
            front = newJob;
        }
    }

    void dequeue() {
        if (isEmpty()) {
            cout << "Queue is empty" << endl;
            return;
        }
        Job* temp = front;
        front = front->next;
        if (!front) {
            back = nullptr;
        }
        delete temp;
    }

    void display() {
        Job* current = front;
```

```
        while (current) {
            cout << "Job ID: " << current->id << ", User: " << current->user << ", Pages: "
<< current->pages << endl;
            current = current->next;
        }
    }
};

int main() {
    Printer printer;

    // Adding jobs
    printer.enqueue(1, "Alice", 10);
    printer.enqueue(2, "Bob", 5);
    printer.enqueue(3, "Charlie", 20);

    cout << "Current printer queue:" << endl;
    printer.display();

    // Processing jobs
    cout << "\nProcessing jobs..." << endl;
    printer.dequeue();
    printer.display();

    return 0;
}
```

```
Current printer queue:
Job ID: 1, User: Alice, Pages: 10
Job ID: 2, User: Bob, Pages: 5
Job ID: 3, User: Charlie, Pages: 20

Processing jobs...
Processing Job ID: 1, User: Alice, Pages: 10
Processing Job ID: 2, User: Bob, Pages: 5
Processing Job ID: 3, User: Charlie, Pages: 20
```

Using the exercise we did in the STL as a guide, this is the output that I have made with a few changes in the int main () to make it more presentable, The output is the way it is because it involves enqueuing and dequeuing once one user is done using queuing with a linked list.

8. Conclusion

9. Assessment Rubric