CADT
IDT វិទ្យាស្ថានបច្ចេកវិទ្យាឌីជីថល
Institute of Digital Technology

Course T1Y2: Advanced Algorithms

Lecturer: Bou Channa

Student's name: Chea Ilong

ID: 100022

Group: 1 SE Gen10

Lab 5: Assignment

Exercise1:

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

struct Nodes
{
    int value;
    Nodes *next;
};

class Stack
{
private:
```

```
    int length;
    Nodes *top;

public:
    Stack()
    {
        top = nullptr;
        length = 0;
    }
    ~Stack()
    {
        while (!isEmpty())
        {
            pop();
        }
    }
    void push(int newValue)
    {

        Nodes *newNode;
        newNode = new Nodes;

        newNode->value = newValue;
        newNode->next = top;
        top = newNode;
        length++;
```

```
};
void pop()
{
    if (length == 0)
    {
        cout << "The stack is empty";
    }
    else
    {

        Nodes *temp = top;
        top = top->next;
        delete temp;
        length--;
    }
};
int peek()
{
    if (length == 0)
    {
        cout << "The stack is empty";
        return -1;
    }
    else
    {
        return top->value;
```

```
        }
    };
    bool isEmpty()
    {
        return (length == 0);
    };
    string print()
    {
        Nodes *t = top;
        string store = " ";
        while (t != nullptr)
        {
            store += to_string(t->value) + " ";
            t = t->next;
        }
        return store;
    }
    int size()
    {

        return length;
    };
};
```

```
Stack:
Test 1:
Stack contents:  44 33 22 11
Stack size: 4
Test 2:
Stack contents after pop:  33 22 11
Stack contents after pop:  22 11
Stack size: 2
Is stack empty? No
Peeking 22
Test 3:
Stack contents after pop x 2
Is stack empty? Yes
```

Exercise2: Queue

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

struct Node
{
    int value;
    Node *next;
};

class Queue
{
private:
    Node *front, *rear;
```

```
    int length;

public:
    Queue()
    {
        front = nullptr;
        rear = nullptr;
        length = 0;
    };
    ~Queue()
    {
        while (!isEmpty())
        {
            dequeue();
        }
    };

    void enqueue(int enter)
    {
        Node *newNode;
        newNode = new Node;
        newNode->value = enter;
        newNode->next = nullptr;

        if (length == 0)
        {
```

```
            front = newNode;
            rear = newNode;
        }
        else
        {
            rear->next = newNode;
            rear = newNode;
        }
        length++;
    };
    void dequeue()
    {

        if (length == 0)
        {
            cout << "The Queue is empty";
        }
        else
        {
            Node *temp = front;
            front = front->next;
            delete temp;
            length--;
        }
    };
    int peek()
```

```
{
    if (length == 0)
    {
        cout << "The Queue is empty";
        return -1;
    }
    return front->value;
};
bool isEmpty()
{
    return (length == 0);
};
string print()
{
    Node *t = front;
    string store = " ";
    while (t != nullptr)
    {
        store += to_string(t->value) + " ";
        t = t->next;
    }
    return store;
};
int size()
{
    return length;
```

```
    };
};
```

```
Queue:
Test:
 11 22 33
 22 33
Front of the queue is: 22
Queue is empty: false
Queue size is: 2
PS C:\Users\MSI PC\Desktop\lab5>
```

Test case:

```
#include "Stack.h"
#include "Queue.h"
#include <string>
#include <iostream>

using namespace std;

int main()
{

    // Test cases for stack class
    Stack mystack;
    cout << "Stack: " << endl;
```

```
    cout << "Test 1: " << endl;
    mystack.push(11);
    mystack.push(22);
    mystack.push(33);
    mystack.push(44);
    cout << "Stack contents: " << mystack.print() <<
endl;
    cout << "Stack size: " << mystack.size() << endl;
    cout << "Test 2: " << endl;
    mystack.pop();
    cout << "Stack contents after pop: " <<
mystack.print() << endl;
    mystack.pop();
    cout << "Stack contents after pop: " <<
mystack.print() << endl;
    cout << "Stack size: " << mystack.size() << endl;
    cout << "Is stack empty? " << (mystack.isEmpty()
? "Yes" : "No") << endl;
    cout << "Peeking " << mystack.peek() << endl;
    cout << "Test 3: " << endl;
    mystack.pop();
    mystack.pop();
    cout << "Stack contents after pop x 2" << endl;
    cout << "Is stack empty? " << (mystack.isEmpty()
? "Yes" : "No") << endl;
```

```
    // Test cases for Qeueu class
    Queue myQueue;
    cout << "Queue: " << endl;
    cout << "Test: " << endl;
    myQueue.enqueue(11);
    myQueue.enqueue(22);
    myQueue.enqueue(33);
    cout << myQueue.print() << endl;
    myQueue.dequeue();
    cout << myQueue.print() << endl;
    cout << "Front of the queue is: " <<
myQueue.peek() << endl;
    cout << "Queue is empty: " << boolalpha <<
myQueue.isEmpty() << endl;
    cout << "Queue size is: " << myQueue.size() <<
endl;



    return 0;
}
```

```
Stack:
Test 1:
Stack contents:  44 33 22 11
Stack size: 4
Test 2:
Stack contents after pop:  33 22 11
Stack contents after pop:  22 11
Stack size: 2
Is stack empty? No
Peeking 22
Test 3:
Stack contents after pop x 2
Is stack empty? Yes
Queue:
Test:
 11 22 33
 22 33
Front of the queue is: 22
Queue is empty: false
Queue size is: 2
PS C:\Users\MSI PC\Desktop\lab5>
```

Exercise4:

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

struct Student
{
    int ID;
    std::string name;
```

```
    int phone_number;
    std::string gender;
    std::string major;
    Student *next;
};

class Queue
{
private:
    Student *front, *rear;
    int length;

public:
    Queue()
    {

        front = nullptr;
        rear = nullptr;
        length = 0;
    };
    ~Queue()
    {
        while (!isEmpty())
        {
            dequeue();
        }
```

```
};

void enqueue(int ID,
             string name,
             int phone_number,
             string gender,
             string major)
{
    Student *newNode;
    newNode = new Student;
    newNode->name = name;
    newNode->ID = ID;
    newNode->phone_number = phone_number;
    newNode->gender = gender;
    newNode->major = major;
    newNode->next = nullptr;

    if (length == 0)
    {
        front = newNode;
        rear = newNode;
    }
    else
    {
        rear->next = newNode;
        rear = newNode;
```

```
        }
        length++;
    };
    void dequeue()
    {

        if (length == 0)
        {
            cout << "The Queue is empty";
        }
        else
        {
            Student *temp = front;
            front = front->next;
            delete temp;
            length--;
        }
    };
    Student *peek()
    {
        if (length == 0)
        {
            cout << "The Queue is empty" << endl;
            return nullptr;
        }
        return front;
```

```
};

bool isEmpty()
{
    return (length == 0);
};
void print()
{
    if (isEmpty())
    {
        cout << "The Queue is empty" << endl;
    }
    Student *t = front;
    while (t != nullptr)
    {
        cout << "ID: " << t->ID
             << ", Name: " << t->name
             << ", Phone: " << t->phone_number
             << ", Gender: " << t->gender
             << ", Major: " << t->major << endl;
        t = t->next;
    }
}
void print3()
{
    if (isEmpty())
```

```
        {
            cout << "The Queue is empty" << endl;
        }
        Student *t = front;
        int count = 0;
        while (t != nullptr && count != 3)
        {

            cout << "ID: " << t->ID
                 << ", Name: " << t->name
                 << ", Phone: " << t->phone_number
                 << ", Gender: " << t->gender
                 << ", Major: " << t->major << endl;
            count++;
            t = t->next;
        }
    };
    int size()
    {
        return length;
    };
};

int main()
{
    Queue myqueue;
```

```
    int choice;
    do
    {
        cout << "\nQueue Menu:" << endl;
        cout << "1. Enqueue student" << endl;
        cout << "2. Dequeue student" << endl;
        cout << "3. Print all students" << endl;
        cout << "4. Print first 3 students" << endl;
        cout << "5. Check queue size" << endl;
        cout << "6. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice)
        {
        case 1:
        {
            int ID, phone_number;
            string name, gender, major;

            cout << "Enter student ID: ";
            cin >> ID;
            cin.ignore();

            cout << "Enter student name: ";
            getline(cin, name);
```

```
            cout << "Enter student phone number: ";
            cin >> phone_number;

            cout << "Enter student gender: ";
            cin >> gender;

            cout << "Enter student major: ";
            cin.ignore();
            getline(cin, major);

            myqueue.enqueue(ID, name, phone_number,
gender, major);
            cout << "Student added to the queue." <<
endl;
            break;
        }
        case 2:
            myqueue.dequeue();
            break;
        case 3:
            myqueue.print();
            break;
        case 4:
            myqueue.print3();
            break;
```

```
        case 5:
            cout << "Queue size: " << myqueue.size()
<< endl;
            break;
        case 6:
            cout << "Exiting program..." << endl;
            break;
        default:
            cout << "Invalid choice. Please try
again." << endl;
        }

    } while (choice != 6);

    return 0;
}
```

```
6. Exit
Enter your choice: 3
ID: 1, Name: long, Phone: 123456789, Gender: m, Major: computer science
ID: 2, Name: ling, Phone: 12345678, Gender: f, Major: software engineering
ID: 3, Name: visa, Phone: 12345678, Gender: m, Major: game dev
ID: 4, Name: nuth, Phone: 12345678, Gender: m, Major: mechanical engineering
ID: 5, Name: joe, Phone: 123456789, Gender: m, Major: data science
ID: 6, Name: goner, Phone: 3456789, Gender: m, Major: digital business
ID: 7, Name: mean, Phone: 98765432, Gender: f, Major: house keeper
ID: 7, Name: joker, Phone: 98765432, Gender: m, Major: TN
ID: 8, Name: can, Phone: 987653456, Gender: f, Major: cyber
ID: 9, Name: chealean, Phone: 3456789, Gender: f, Major: computer science

Queue Menu:
1. Enqueue student
2. Dequeue student
3. Print all students
4. Print first 3 students
5. Check queue size
6. Exit
Enter your choice: 4
ID: 1, Name: long, Phone: 123456789, Gender: m, Major: computer science
ID: 2, Name: ling, Phone: 12345678, Gender: f, Major: software engineering
ID: 3, Name: visa, Phone: 12345678, Gender: m, Major: game dev
```