CPSC 4110/5110/7110

Quantum Computing

Fall 2018

Course Project
Due: TBA

This project aims to deepen your understanding of the course material. Although we simulate a quantum computer using classical algorithms, it will help us appreciate the beauty and capability of a quantum computer.

There are three parts in this description: *undergraduate project*, *graduate project*, and *the common part for both undergraduates and graduates*.

It is suggested that you read the common part first, and then go to the respective project part you need to do.

It is a group project and each group can have up to four members. Each member will be awarded the same mark.

(1) Undergraduate project

(a) Implement a Controlled-NOT as a function and test it.
(b) Implement a Toffoli gate as a function and test it.
(c) Implement Deutsch's algorithm and test it.

So you need to create three programs, called *cnot*, *toffoli*, and *deutsch*, respectively, and test them.

(2) Graduate Project

(a) Implement a Controlled-NOT as a function and test it.
(b) Implement a Fredkin gate as a function and test it.
(c) Implement Deutsch's algorithm and test it.
(d) Implement Grover's search algorithm and test it.

So you need to create three programs, called *cnot*, *fredkin*, *deutsch*, and *grover* respectively, and test them.

(3) The common part

(a) You need to prepare a 15-minute demo to show that your programs work. The time and location of the demo will be determined shortly.

(b) You need to consider how to implement a quantum state using classical data structures. Write and submit a report up to 4 pages, in which you should document your key data structures, diagrams, etc. The due of the report will be determined shortly.

(c) The required language to do this project is C++, which is the one we teach in our department.

(d) You may find the functions on the following list are useful. Note that some later functions on the list can be implemented using some earlier functions. Of course, you can create any other help functions, if needed.

- Write a function that accepts two complex numbers and returns their sum and product.
- Write a function that accepts one complex number and returns its modulus and conjugate.
- Write a function that accepts *n*-dimension complex vectors and returns addition and dot product.
- Write a function that accepts two appropriate complex matrices and returns addition and multiplication.
- Write a function that accepts a square complex matrix and returns true if it is a Hermitian or false otherwise.
- Write a function that accepts two complex matrices and returns their tensor product.
- Write a function that implements an observable: the arguments of the function are a square complex matrix and a ket, the function checks whether the matrix is a Hermitian, and if it is, the function calculates the expected value of the observable on the ket.
- Write a function that implements one step in an orbit: the arguments of the function are a square complex matrix and a ket, the function checks whether the matrix is a unitary, and if it is it, the function calculates the result of applying the matrix to the ket.
- Write three functions called X, Y, and Z, respectively, to simulate the Pauli matrices.
- Write a function called H to simulate the Hadamada matrix.
- ……

(e) Copy your source code for each program into a separate folder on a USB stick/CD. Submit it to me on the due date. It will be returned back to you after the project is marked.

(f) There might be other parts if necessary that will be announced shortly.