COMP SCI 4413, 4813, 7413	Introduction to Quantum Computing

Assignment 2

Author: Yasir Latif (S2 2019), Tat-Jun Chin (S1 2020)

Submit on MyUni before 11:59pm 30 April 2020

Instructions

- Attempt all sections. If necessary, refer to Rieffel and Polak [RP] for more information.
- The deliverables include a program (Section 1) and a report (Section 2) to be submitted via MyUni. You will be evaluated on both the code as well as the report.

1 Quantum circuit simulator - 50 marks

In either C/C++, Java or Python, implement a simple quantum simulator. If C/C++, you must supply a makefile (named exactly as `Makefile`) with a rule called `quantum` to compile your program into a Linux executable binary named `quantum.bin`. Your program must be able to be compiled and run as follows:

```
$ make quantum
$ ./quantum.bin [circuit]
```

If Java, write your program in the file `quantum.java`. Your program must be able to be compiled and run as follows:

```
$ javac quantum.java
$ java quantum [circuit]
```

If Python, write you program in the file `quantum.py`. Your program must be able to be run as follows:

```
$ python quantum.py [circuit]
```

Make sure that your Python program is compatible with Python 3.

The input to your program is a text file specified by the path `[circuit]`. Take note that

- I do not have a Windows machine and it will be difficult for me to locate one; thus
- I will be running your program on a Unix-like environment (e.g., Linux, Mac OS X) and the path `[circuit]` will follow the Unix convention.

If you do not have a Unix-like machine, consider dual booting your machine with Linux or use a simulation environment such as Cygwin for Windows.

In this assignment, we will only consider two-qubit systems. The input text file pointed to by `[circuit]` contains the design of a quantum circuit and desired input quantum states. The specific formatting of the file is as follows:

```
T M
L11 L12 ... L1T
L21 L22 ... L2T
Q11 Q12 Q13 Q14
Q21 Q22 Q23 Q24
.   .   .   .
.   .   .   .
.   .   .   .
QM1 QM2 QM3 QM4
```

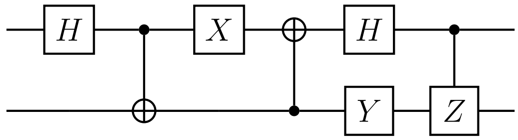
More details:

- The first line contains the number `T` of quantum transformations performed by the circuit (`T` is the number of "columns" or steps in the circuit), and the number `M` of input quantum states to process.
- The subsequent two lines correspond to the two "wires" in the circuit. Each character `Lnt` (e.g., `L25` is the character at the 2nd wire and 5th quantum transformation) can be one of the following symbols:
  - Identity (single qubit).
  - `X` Not (single qubit).
  - `Z` Phase shift (single qubit).
  - `Y` Negation and phase shift (single qubit).
  - `H` Hadamard (single qubit).
  - **.** Control bit.
  - \+ Target bit.
- The subsequent M define the input quantum states in linear algebra (vector) form, where

  ```
  Qm1 Qm2 Qm3 Qm4
  ```

  is the m-th ($1 \leq m \leq M$) input quantum state.

Example: For the quantum circuit below,



where we wish to evaluate the following input states,

- $|v\rangle = |11\rangle$
- $|v\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle$
- $|v\rangle = 0.7555\,|00\rangle + 0.7584\,|01\rangle + 0.3065\,|10\rangle + 0.7606\,|11\rangle$

the corresponding input text file contains

```
6 3
H.X+H.
-+-.YZ
0 0 0 1
0.7071 0 0.7071 0
0.7555 0.7584 0.3065 0.7606
```

□

Given the path `[circuit]` to an input file in the correct format, your program should print to standard output ("the command line") the output quantum states obtained by processing the input states using the given quantum circuit. Specifically, for the input file that contains

```
T M
L11 L12 ... L1T
L21 L22 ... L2T
Q11 Q12 Q13 Q14
Q21 Q22 Q23 Q24
.   .   .   .
.   .   .   .
.   .   .   .
QM1 QM2 QM3 QM4
```

your program should print to standard output

```
S11 S12 S13 S14
S21 S22 S23 S24
.   .   .   .
.   .   .   .
.   .   .   .
SM1 SM2 SM3 sM4
```

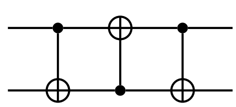
where

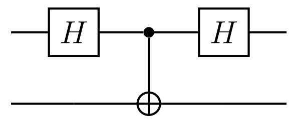
```
Sm1 Sm2 Sm3 Sm4
```

is the output quantum state for input quantum state

```
Qm1 Qm2 Qm3 Qm4
```

Self testing Verify that your program works - sanity test using the following simple circuits.





Submission Submit your program files in a zip package named `assignment2.zip` on MyUni.

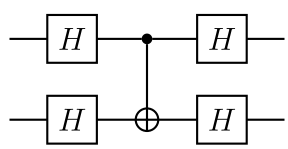
Evaluation I will be testing your program using several quantum circuits to verify its correctness. If your program gives the correct output to all tests, you will receive all the marks.

2 Testing your simulator

Answer all questions (and subquestions) below in a report named `assignment2.pdf` to be submitted via MyUni.

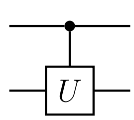
Question 1 Equivalent gate (10 marks)

What does this following circuit do? Check using your quantum simulator.



Question 2 Inverting control bit (10 marks)

The following circuit applies transformation U on the second qubit if the first qubit is in state 1.



Design a circuit that applies transformation U on the second qubit if the first qubit is in state 0. Verify using your quantum simulator.

Question 3 Generating EPR pairs (10 marks)

Define a circuit that can create EPR pairs. This circuit takes as input $|00\rangle$ and produces $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Verify the correctness of your circuit using your simulator.

- What happens if the input is not $|00\rangle$? Is the output state of your circuit still entangled? Briefly explain your answer.

Question 4 Dense coding (20 marks)

Using your simulator, implement a circuit to perform dense coding as described in the lectures. The circuit takes $|00\rangle$ as input, creates an entangled pair (Question 1) and then perform encoding. The second part of the circuit decodes the incoming bits to extract two classical bits.

- How critical is the choice of input to this circuit? Does the scheme fail if the input changes to, say, $|11\rangle$?

$\Longleftarrow$ End of file $\Longrightarrow$