G53NSC and G54NSC
Non-Standard Computation
Lab 6 Exercises

Dr. Alexander S. Green

11th March 2010

Exercise sheet 6

These exercises carry on from Exercise sheets 1, 2, 3, 4 and 5; and may use some of the types and functions you have previously defined.

The exercises are listed here, but more information can be found in the hints and tips section below.

1. Implement, using *QIO*, instances of Grover's algorithm for search spaces of size $N = 4$, $N = 8$, and $N = 16$.

2. Shor's algorithm requires a unitary that can be used to calculate the function $f(x) = b^x (mod\ \ n)$ over a quantum state. We could extend on the reversible adder we defined for exercise sheet 2 to build a library of reversible arithmetic functions that can be used in *QIO*. Using the unitary defined for reversible addition in exercise sheet 2, define a unitary for reversible modular addition.

3. Assume you have a function $period :: Integer \rightarrow Integer \rightarrow QIO\ Integer$, such that $period\ n\ b$ is a quantum computation that returns the period of the function $f(x) = b^x (mod\ \ n)$, write an effectful classical computation that uses the *period* function to calculate the factors of a large number N, defined such that $N = pq$, where p and q are different prime numbers.

Hints and Tips

Exercise information

1. For each size of search space, you will need to define a unitary that corresponds to V, a unitary that corresponds to W, and calculate how many iterations of these are required. The unitary V depends upon an input search function, e.g. for $N = 8$, this function would have type

search :: $(Bool, Bool, Bool) \rightarrow Bool$, and only return *True* for the single element that is being searched for. V can be defined using the trick of an ancilliary qubit that is set to $|-\rangle$. W is hard to define, so you may wish to define $-W$ instead, which doesn't effect the measurement probabilities. $-W$ doesn't depend on the input *search* function. Test your solutions by defining some instances of the *search* functions, and simulating the running of the computation to check the probabilities of measuring the correct solution. Remember, for small N (as is the case here), you will need to take the original angle θ of the $|\phi\rangle$ state into account when calculating the number of iterations. Try and show your workings for calculating the angle θ as comments in your code, or you could define a function in Haskell that calculates the number of iterations for you.

2. I would suggest reading the paper "Quantum networks for elementary arithmetic operations" that is linked from the module web page. Feel free to implement more of the arithmetic functions if you are feeling adventurous.

3. You may find it useful to implement a dummy function for *period* so that your programs still type check correctly. E.g.

   period :: *Integer* $\rightarrow$ *Integer* $\rightarrow$ *QIO Integer*
   period b n = *return* 0

   You can think of the *period* function as the quantum part of Shor's algorithm. What you have to define, are the classical parts of the computation that turn factorisation into a period finding problem. That is, you need to define an overall function *factorise* :: *Integer* $\rightarrow$ *IO* (*Integer*, *Integer*), that randomly picks a value for b that is co-prime to the value of n, calls the *period* function with these values, and then checks to see if the returned value has the two necessary properties. If it has, then you can use it to calculate the two return results of the factorisation function, otherwise you need to restart the algorithm with a new random value for b.

   As the *period* function is only a dummy function (E.g. it doesn't return the actual period of the function $f(x) = b^x (mod\ \ n)$) you won't be able to test your solution directly. If you're feeling adventurous, then you may wish to define the *period* function (either (unefficiently) classically or quantumly), or have the dummy value it returns set to a suitable value for some test input.

The Quantum IO Monad

The Quantum IO Monad, or *QIO* is a monadic interface from Haskell to quantum computation. More precisely, it is a library that allows you to define unitary operators and effectful quantum computations, along with simulator functions that allow you to *run* the quantum computations that you define. A lot of information on *QIO* including its implementation are available online (see the links on the course webpage). Installation of *QIO* is relatively straightforward

if you can make use of cabal (cabal is part of the Haskell platform, and as such should already be installed on the machines in A32).

The following list of instructions will install *QIO* on the windows machines in A32 (but you may need to re-install it for every session). The following commands should be entered in a command prompt:

- Set the http proxy in the current command prompt

```
set HTTP_PROXY=wwwcache.cs.nott.ac.uk:3128
```

- Make sure the cabal list of packages is up to date:

```
cabal update
```

- Install *QIO* (in your own user space, as you don't have global permissions)

```
cabal install QIO --user
```

(note: if you don't have a proxy, and you are using your own machine, then you should just have to update the list of packages as above, and install the *QIO* package without the –user flag)

If you are having difficulties installing *QIO* you can always download the source from: http://www.cs.nott.ac.uk/ asg/QIO/ and import the files as necessary. However, i would recommend this as a last resort, and suggest that you contact me for support.

Information

The exercises set in the labs have a firm deadline of 12:00 (midday); Thursday the 1st of April, but it is highly recommended that you submit your work on a weekly basis (E.g. 1 week after the date each exercise sheet is released) to enable you to receive ongoing feedback. I will give feedback for any exercises submitted within 2 weeks of their original release date.

The weekly submissions should be emailed to me (asg@cs.nott.ac.uk), or handed to me in the labs. The final submission of your portfolio will be through the school office by 12:00 (midday) on Thursday the 1st of April (The last day of the Spring term). The final submission through the school office should be made even if you have been submitting work to me on a weekly basis as it is this final submission that counts as your portfolio.

These exercise sheets should be attempted on your own, and at the end of the course, it is these individual submissions that will make up your portfolio project. Combined, the work submitted in your portfolio is worth 50% of the mark for this module (The other 50% consisting of the research report and presentation).