COMP1927 15s2 Final Exam

Question 1 (10 marks)

In the `q1` directory (in the file `main.c`), is a program which

- reads a sequence of integers from stdin and stores them in a binary search tree `t`
- displays the tree
- calls a function `map(t,f)` to modify the value in each node in the tree using a function parameter `f`
- displays the modified tree

It determines which function to apply based on a command line argument. The following examples show which functions are available:

Command	Effect
`./q1 1 < tests/tree1`	reads values from file `tests/tree1` and decrements each value by 1 (using the function `decr()`)
`./q1 2 < tests/tree1`	reads values from file `tests/tree1` and doubles each value (using the function `duble()`)
`./q1 3 < tests/tree1`	reads values from file `tests/tree1` and squares each value (using the function `square()`)

The `decr()`, `duble()` and `square()` functions are all defined in `main.c`. The `map()` function is defined in the `BSTree` ADT.

The `q1` directory also contains an implementation of a binary search tree ADT in the files `BSTree.h` and `BSTree.c`. You do not need to understand the details of all of the functions in `BSTree.c`, but you should familiarise yourself with the structure of `BSTreeNode`s as defined in that file.

Your task for this question is to implement the `map()` function in the `BSTree.c` file.

The `map()` function is defined as follows:

```
void map(BSTree t, int (*f)(int)) { ... }
```

and takes two parameters:

- `t` ... an instance of the `BSTree` ADT (i.e. a binary search tree)
- `f` ... a pointer to a function mapping an `int` to an `int`

The `map()` function applies `f` to the value in each tree node, and then replaces the original value in the node by the result returned by the function. The actual function supplied for `f` must have the type `int→int` (e.g. like `decr()`). You can find a skeleton `map()` function at around line 137 in the `BSTree.c` file.

You can find out more about the behaviour of the `q1` program by looking at the files in `q1/tests` directory. Each file named `tX.sh` contains the commands to run one test. Each test uses one of the files named `treeX` as input. Each test has a corresponding file `tX.exp` which contains the expected output from a correct implementation of `q1`, run using `tX.sh`.

The `q1` directory also contains a `Makefile` which you use as:

```
make q1     # build the q1 program
```

You can test your `q1` program using the command:

```
check q1    # run tests on the q1 program
```

Once you are satisfied with your program, submit it using the command:

```
submit q1
```

This will make a copy of the `BSTree.c` file from the `q1` directory as your answer for this question. You can run the `submit` command as many times as you like, but make sure that your final submission compiles without any errors or warnings. Test your program thoroughly, possibly using test cases additional to those supplied. Your program will be tested using inputs which are different to the examples in the `q1/tests` directory.

You can add any additional functions (apart from `map()`) to the `BSTree.c` file, but you may not change any of the other files.

If, at some stage, you need to "re-install" the files (although you should not need to), you can copy all of the original files into the `q1` directory by running the command:

```
re-start q1
```

Beware: this will overwrite all of your exsting files for this question, so only do it if you seriously mess things up.