1. (Rebalancing)

   Complete this week's Binary Search Tree ADT (`BST.h`, `BST.c`, also needs `queue.h`, `queue.c`) from the lecture by an implementation of the function:

   ```
   Tree rebalance(Tree t) { ... }
   ```

   *We have created a script that can automatically test your program. To run this test you can execute the* `dryrun` *program that corresponds to this exercise. It expects to find the file named* `BST.c` *in the current directory with your implementation of the function* `rebalance()`.

   You can use dryrun as follows:

   ```
   prompt$ 9024 dryrun BST
   ```

2. (Red-black trees)

   Consider the following high-level description from the lecture of an algorithm for inserting items into a red-black tree:

   ```
   insertRB(tree,item,inRight):
      if tree is empty then
         return newNode(item)
      else if item=data(tree) then
         return tree
      end if
      if left(tree) and right(tree) are RED then
         split 4-node
      end if
      recursive insert, re-arrange links/colours after insert
      return modified tree

   insertRedBlack(tree,item):
      tree=insertRB(tree,item,false)
      colour(tree)=BLACK
      return tree
   ```

   Implement this algorithm in the Red-Black Tree ADT (`RBTree.h`, `RBTree.c`) from the lecture as the function:

   ```
   Tree TreeInsert(Tree t, Item it) { ... }
   ```

   *We have created a script that can automatically test your program. To run this test you can execute the* `dryrun` *program that corresponds to this exercise. It expects to find the file named* `RBTree.c` *in the current directory with your implementation of the function* `TreeInsert()`.

   You can use dryrun as follows:

   ```
   prompt$ 9024 dryrun RBTree
   ```

## Assessment

Submit your solutions using the following **give** command:

```
prompt$ give cs9024 week8 BST.c RBTree.c
```

Make sure you spell the filenames correctly. You can run **give** multiple times. Only your last submission will be marked.

The deadline for submission is **Tuesday, 28 July 11:00:00am**.

Each program is worth 1 mark. Total marks = 2. Auto-marking will be run by the lecturer several days after the submission deadline using different test cases than `dryrun` does.

*Hints*:

- Programs will not be manually marked.
- It is important that the output of your program follows exactly the format shown above, otherwise auto-marking will result in 0 marks.
- **Ensure that your program compiles on a CSE machine with the standard options `-Wall -Werror -std=c11`. Programs that do not compile will receive 0 marks.**
- `dryrun` and auto-marking also check the correct usage of dynamic memory allocation and pointers as well as the absence of memory leaks.
- Do your own testing in addition to running `dryrun`.

## Plagiarism

Group submissions will not be allowed. Your programs must be entirely your own work. Plagiarism detection software will be used to compare all submissions pairwise (including submissions for similar assessments in previous years, if applicable) and serious penalties will be applied, including an entry on UNSW's plagiarism register.

- ***Do not copy ideas or code from others***
- ***Do not use a publicly accessible repository or allow anyone to see your code***

Please refer to the on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Plagiarism and Academic Integrity
- UNSW Plagiarism Policy Statement
- UNSW Plagiarism Procedure