What time is it for you?

This is an individual assignment. Due: Sunday 27th August 11:59:59pm (end of Week 5)

Your task for the first assignment (Assignment 0) is to write a program that will **work out what the local time is in your city**, to help Andrew get the timezone conversion correct so that you don't get woken up at 3am with a phone call.

To do this, you will write a function to calculate the local time in a given city (in Australia or New Zealand), given the current time in UTC.

Keep reading for more details.

Background

Here's the situation:

After the course is finished, when everything's all done, and all of your results are out, and everything's been great, I'm going to call the student who had the most fantastic final assignment.

I'll call them up, and I'll say: "Hello [student's name here], your assignment was really fantastic, I just wanted to let you know, it was amazing, I'm so proud of you".

But, it's hard for me to do that – because you might be travelling around the country, or I might be travelling around the country, or I might be back in New Zealand, and I don't want to call you in the middle of the night; as much as you want me to say "hey, your assignment was great!', you probably don't want me to say that at 3 in the morning, and wake you up.

So, your task for this first assignment is to **help me out by writing a program that will work out what the time is for you, when I'm trying to call you.**

So I can use that to go "Is this going to be an okay time to call you? It's this time for you in your local timezone, so yes I can or no I can't call you".

And so, the way that the assignment is going to work: **we'll give you a city (and we'll limit it to Australia and New Zealand, to keep it simpler). We'll also give you the date (day and month, the year will always be 2017) and the time in UTC.**

As an example…

If I gave you the day is the 9th, the month is August, the time in UTC is 6:30am, and the location is Sydney, tell me the time in Sydney.

The timezone Sydney is currently in – it's not the summer, so we're in Australian Eastern Standard Time (AEST), which is UTC+10. So, if you added 10 hours onto UTC time, 6:30am + 10 hours gives 4:30pm.

Note: You don't have to worry about actually getting the time and scanning it in and parsing it yourself – we'll give your function the number for the day and the month and the time, and you've just got to give us a number back – the time in that city's local time.

Summary

Take the stub code below and finish implementing the `getLocalTime()` function it contains. The main function provided runs a series of unit tests on your function to see if it contains an error.

Why *this* assignment?

There are three reasons we chose this particular task for your first assignment:

The first is to get you to work out how to compute a function using ifs which is non-trivial. It's not hard but it does require you to think carefully. Or rather - once you've solved it won't seem hard at all, but it likely will seem tricky until you work it out.

The second is to get you into the habit of planning your programs with pen and paper before you start trying to write them. First make sure you understand the problem, then plan out how your program might solve it drawing pictures and diagram. Only then start typing. You should spend much more time planning and understanding the problem than you do typing.

The final objective of this task is to get you get you used to the process of writing a successful program. There are two parts to this - both of which you might find surprising if you have not done any programming before now.

One is the need to experiment, to try, and not to fear or be daunted by (repeated) failure. If you persevered to the end of [GrowCube] then you have demonstrated that you already have what it takes, otherwise you should set as a personal goal to work at developing your perseverance and courage and determination.

The other central part of writing a good program is the need to be skeptical of the correctness of what you produce. In this task we introduce the concept of unit tests. We'll be developing this idea over the rest of the course and it will be a central part of everything you do from now on.

Tips for getting started

Don't worry so much about the syntax of C at first, or what you will type in. Instead the most important thing to do at the start of any program is to make sure you understand the problem well. Play around with it. Do examples on pen and paper. Draw diagrams. Don't be in a rush to start typing.

Finally, don't be afraid to ask questions if you get stuck or need any help.

Unit Tests

Unit tests are a series of small tests which test self-contained parts ("units") of the programs you write. In this task the unit you are testing is the `getLocalTime()` function.

Unlike the other activities you've submitted up until now we'll only mark your `getLocalTime()` function at the end of the submission period. Furthermore when we mark your submission we'll use our own tests and you won't know what these are in advance. So the only way you will know if your function is correct before we mark it is to run your own unit tests on it. The sample code provided below contains a number of demonstration unit tests to show you what they look like and to give you a feel for them. They are not an extensive set of tests. You will need to add to them.

It is fine to collaborate and to share unit tests among yourselves. Build up a shared suite of unit tests on the page below. Everyone should aim to add at least one test to the collection.

Contribute to the unit tests
Add at least one test to the assignment unit tests.

How long should you spend on this assignment?

Experienced programmers should set themselves a challenge to complete this task before the end of the week. We'll count the number of times you submit before your program is correct. As a personal goal you should aim to have your first submission to be correct.

New programmers should allow up to two weeks to get it all working perfectly. Start at once and set some time aside to work on it regularly. You will not solve it well in one long intense session. Instead plan regular periods. Don't be afraid to devote more time to it than you expect if it is your first program - the lessons you learn from engaging with it and then succeeding will be invaluable over the rest of the course.

What you are to do

1. Copy the below incomplete program stub into a file on your computer called `localTime.c` .
2. Study and understand timezone conversion. [see the course website for more resources]
3. Contribute at least one good test to the shared unit tests.
4. Edit your copy of `localTime.c` and insert your own code to make it work.
5. When you are confident that your code is correct by running it at home against an appropriate set of unit tests then submit it below. (leave all the tests you used in your submitted code so we can see how carefully you tested...)

Restrictions

- Follow the style guide,
- Don't use syntax features not yet covered,
- Write clear code (make your program a clear and beautiful piece of communication), and
- Don't #include any other files, or change the name or type signature of the `getLocalTime()` function.

Submitting

Write your code in a file called `localTime.c` and submit it below. (make sure you get the file name exactly right, e.g. all lowercase letters etc)

You can comment below if there are any strange messages, or problems with your submission.

Assessment

This assignment will contribute 5% to your final mark.

25% of the marks for assignment 0 will come from hand-marking of the readability of the code you have written. These marks will be awarded on the basis of clarity, commenting, elegance, and style. In other words, your tutor will assess how easy it is for a human to read and understand your program.

1% of the marks for this assignment will come from you having contributed a unit test to the shared unit tests.

The remaining 74% of the marks for this assignment will come from the correctness of your program, i.e. what proportion of the autotests your program passes. These autotests **will not** be available to you during the course of the assignment, and will only be run after the due date of the assignment. Therefore, it is in your best interests to write your own extensive unit tests, so that you can catch any bugs in your program before we run our tests against it.

Late Penalty

The assignment is due at 23:59:59 on Sunday 27th August, Australian Eastern Standard Time.

If your assignment is submitted after this date, then for each hour it is late, the maximum mark it can achieve is reduced by 1%. For example, if an assignment worth 74% was submitted 12 hours late (noon on Monday), the late submission would have no effect. If the same assignment was submitted 30 hours late (6am on Tuesday), the mark would be reduced to 70%, the maximum mark it can achieve at this time.

Skeleton Code

We provide you some *skeleton code* for this assignment, to help you get started.

Download `localTime.c`, or copy it into your current directory on a CSE system by running

```
$ cp /web/cs1511/17s2/assignments/assign0/files/localTime.c .
```

To run some simple automated tests:

```
$ 1511 autotest assign0
```

To run Styl-o-matic:

```
$ 1511 stylomatic localTime.c
Looks good!
```

You'll get advice if you need to make changes to your code.

Submit your work with the *give* command, like so:

```
$ give cs1511 assign0
```

Or, if you are working from home, upload the relevant file(s) to the `assign0` activity on Give Online.