INFT2012 – Application Programming

- Lecturer, tutor (Singapore)
  - Latha Manian
    - email latha.manian@newcastle.edu.au
- Course coordinator (Ourimbah, Australia)
  - Simon
    - email simon@newcastle.edu.au

Your contact details

- The University has given you a NUmail email address
- Use it. If the Uni wants to contact you, it does so via this address
- Technically, Uni staff are not permitted to reply to email apparently from a student, but from some other address
- If you don’t want to use the student email system regularly, set your account to forward to your preferred address . . .
- . . . and if you’re replying to a forwarded message, remember to reply from your NUmail account

Assessment

- Revision quiz, 10%, in class in week 3
- Practical programming test, 20%, in class in week 4
- Programming assignment, 30%, due end of week 11 of classes
  - students are encouraged to work in pairs for this assignment
- Final exam, 40%
  - be sure you're available for the whole of the exam period
- To pass the course, students must attain at least 50% overall, *and* at least 40% on the final exam

No text book

- There is no text book for this course
- You should already know how to program; in this course you will be transferring that knowledge to another programming language and environment, and adding additional topics
- If you like supporting your learning with books, find a book on programming in C#; the one by Bell & Parr is pretty reasonable
- Visual Studio help is very comprehensive, but you need to learn to filter it (eg just to C#); and you will find that some of it is very detailed, for professional developers
- There is also a wealth of support material on the web

Rest of course outline

- It's important to follow some of the links provided in the course outline
- Note the bits about handing in work late, and what to do if your work is affected by illness or other acceptable special circumstances (Application for Adverse Circumstances)
- Check the course Blackboard website weekly for course updates, bulletins, and additional resources
- Note what the outline says about academic integrity, which the university takes very seriously; note that you are required to complete an academic integrity module if you haven't already done so

Plagiarism and other cheating

- The university takes plagiarism and collusion very seriously; students caught plagiarising or colluding can fail the course or worse as a consequence
- In a programming course, this includes . . .
  - using the work of your friends, or working too closely with your friends, in individual work
  - working too closely with other teams in team work
  - getting other people to help too much with your work
  - getting programs from the internet
- If caught, it's not just the copier who is penalised; if you help others too much, or give them copies of your work, you, too, will be penalised, and might fail the course

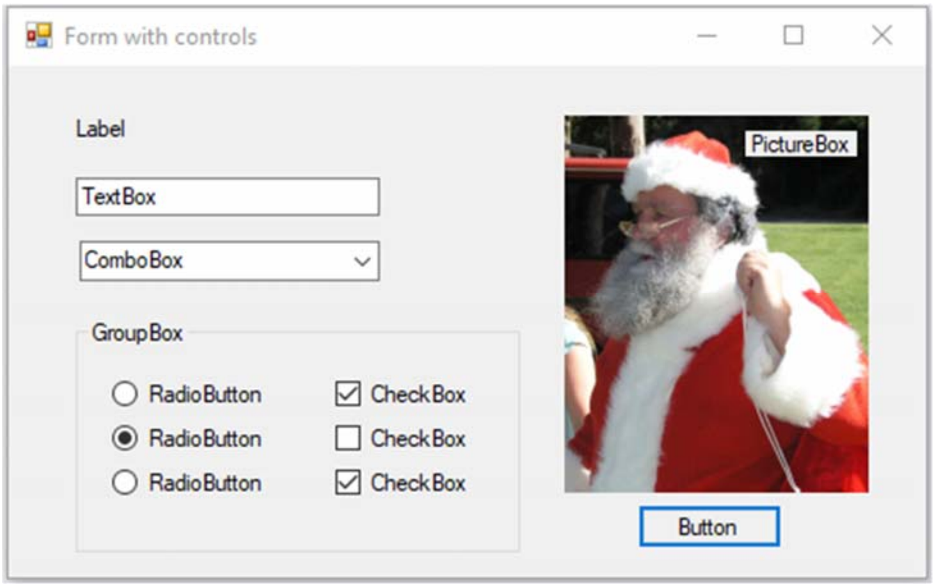
Inft2012 Application Programming

- You already know how to program, because you've completed and passed Inft1004 or its equivalent
- The first thing we'll do in this course is take much of what you already know and translate it to a new language (C#) in a new environment (Visual Studio)
- If you already know C#, this will be revision
- We will then introduce new topics, or go more deeply into topics that were covered only briefly in Inft1004
- By the end of the course you should know more about programming, and you should know a different language and environment

C# and forms

- C# 'programs' are called *solutions*
- A solution can contain a number of *projects*
- In this course, we'll only be writing solutions with single projects, so we can use the words program, solution, and project almost interchangeably
- There are many types of project; all of ours will be Windows Application projects
- Every Windows Application project starts with a form
- We *design* the form, telling it what to look like, by adding controls to it and adjusting their properties
- We *program* the form, telling it how to behave, by adding program code to it

Some common controls on a form



- What's the main difference between radio buttons and check boxes?

Objects have properties

- All objects have *properties*
  - a form has a colour, a size, a title, a position . . .
  - a button has text on it, a text colour, a button colour, a size, a location on the form . . .
- Properties can be set when we are designing the form or in the program code
- A really important aspect of any object is its name

The importance of names

- We want to change the colour of a particular button
- At design time, this is easy – we select the button then change its *BackColor* property
- If we want *the program* to change the colour while it's running, we have to use the button's name to tell the program which button
  - We can't say 'change the colour of the 3rd button from the right'
  - We can say 'change the colour of button *BtnQuit*'
- When we design a form, C# gives default names to controls; we *always* replace these with more helpful names, so that our program code makes sense to us

Objects have methods

- A *method* is a sequence of instructions associated with an object
- A method is a behaviour, an action or sequence of actions defined in program code
- Many methods are already provided by C#
  - eg a form has a method to maximise itself
  - eg a button has a method to hide itself, to apparently disappear from the form it's on
- Other methods can be provided by the programmer

Objects recognise events

- An *event* is something that happens to an object, often as a result of something done by the user
- A control on a form can be clicked, double-clicked, right-clicked, have the cursor pass over it . . . these are all events
- The programmer can write special methods called *event handlers* that tell particular controls how to respond to particular events
- The sort of programming we will be doing is often called *event-driven programming*, because the program does things in response to events such as mouse clicks

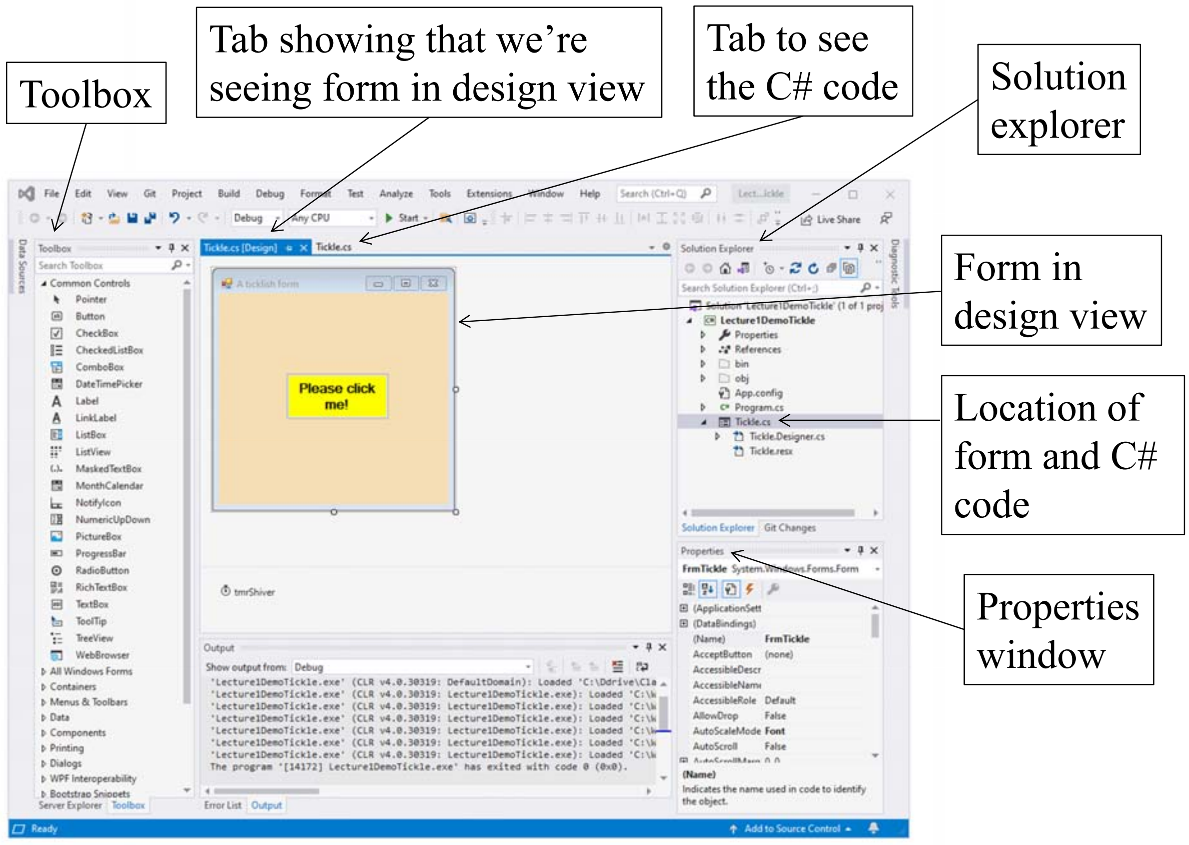
A very simple project

Let's see some of this illustrated in a simple project – *Lecture1DemoTickle*: you will see

- a form in design view
- properties of the form and of the button on it – including their names
- how to change these properties
- methods, named chunks of program code
- event handlers, methods that respond to particular events on particular objects
- variables, assignment statements, and comments
- what happens when you experiment with the code

Visual Studio is cluttered

- The Visual Studio interface can be quite overwhelming until to you get used to it
- On the next slide is an image of the interface for the demo program, with some of the things you might see
- On the following slides are some explanations



Show output from: Debug

```
'Lecture1DemoTickle.exe' (CLR v4.0.30319: DefaultDomain): Loaded 'C:\Ddrive\Cla
'Lecture1DemoTickle.exe' (CLR v4.0.30319: Lecture1DemoTickle.exe): Loaded 'C:\k
'Lecture1DemoTickle.exe' (CLR v4.0.30319: Lecture1DemoTickle.exe): Loaded 'C:\k
'Lecture1DemoTickle.exe' (CLR v4.0.30319: Lecture1DemoTickle.exe): Loaded 'C:\k
'Lecture1DemoTickle.exe' (CLR v4.0.30319: Lecture1DemoTickle.exe): Loaded 'C:\k
'Lecture1DemoTickle.exe' (CLR v4.0.30319: Lecture1DemoTickle.exe): Loaded 'C:\k
'Lecture1DemoTickle.exe' (CLR v4.0.30319: Lecture1DemoTickle.exe): Loaded 'C:\k
The program '[14172] Lecture1DemoTickle.exe' has exited with code 0 (0x0).
```

Properties

FrmTickle System.Windows.Forms.Form

(ApplicationSett	
(DataBindings)	
(Name)	**FrmTickle**
AcceptButton	(none)
AccessibleDescr	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
AutoScaleMode	**Font**
AutoScroll	False

(Name)
Indicates the name used in code to identify the object.

The toolbox

- The toolbox is often tucked away on the left
- If you click it, it expands to the right
- If you want it to stay in view (you normally do when you’re designing a form), click the pin icon to pin it into place
- You’ll normally have it set to Common controls, but there are other options for more advanced use
- From it you can drag controls (eg buttons, labels, texboxes, picture boxes) to the form, then position them and resize them according to your design

Solution explorer

- A solution is the highest-level program; it consists of one or more projects, which consist of many files
- Solution explorer is a kind of file explorer within Visual Studio to help you navigate around these files
- (In Windows explorer, outside Visual Studio, you can see just how many files there are for the simplest Visual Studio program)
- We’re most interested in the .cs (C#) file with the form icon, which contains the form design and the program code written by the user to go with the form

The form in design view

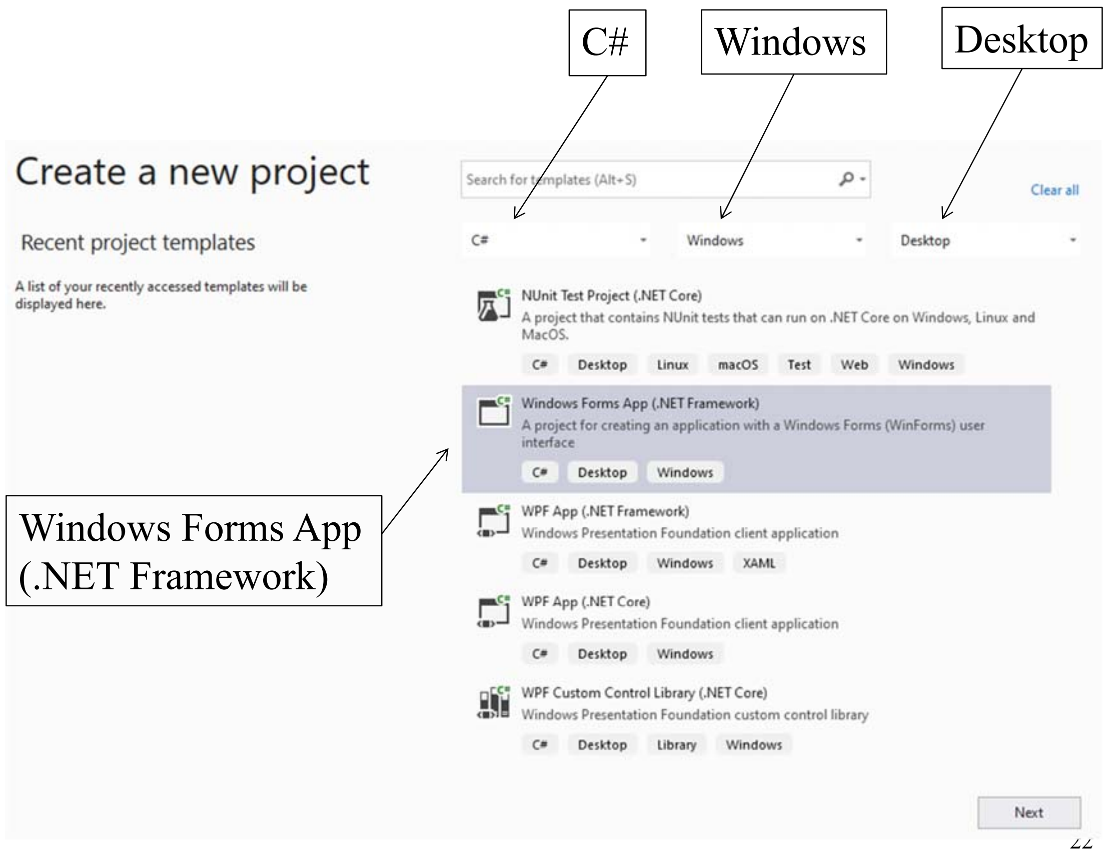
- When you open a new project, you'll be shown an empty form in design view
- When you open an existing project, the form design is often not visible
- To see it, in Solution explorer,
  - double-click on the .cs file name with the form icon next to it
  - or right-click on the same file name and select View designer
  - or click on the file name and press shift-f7

Properties window

- The Properties window is something you'll need open all the time
- If you don't see it, you can get it by clicking the spanner icon near the top of Solution explorer (or select it from the View menu)
- It shows the properties of whatever control is selected
- If the form is selected, it shows the form's properties
- If the button is selected, it shows the button's properties
- You won't use all of the these properties, but you should take a look to see what sort of thing is there
- This is also where you can change properties from their default values

Creating a new project

- Let’s write a program to do some simple drawing
- In Visual Studio, choose File / New / Project
- In the left pane select Visual C#, and in the middle pane select Windows Forms Application
- Be absolutely sure to enter a new and informative name for the project – <u>do not</u> leave it with the name *WinFormsApp1*; call this one *Lecture1Drawing*
- Be absolutely sure to enter a sensible location for the project, if you want to be able to find it again


C#
Windows
Desktop
Windows Forms App (.NET Framework)
Search for templates (Alt+S)
Clear all
C#
Windows
Desktop
Next

Create a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

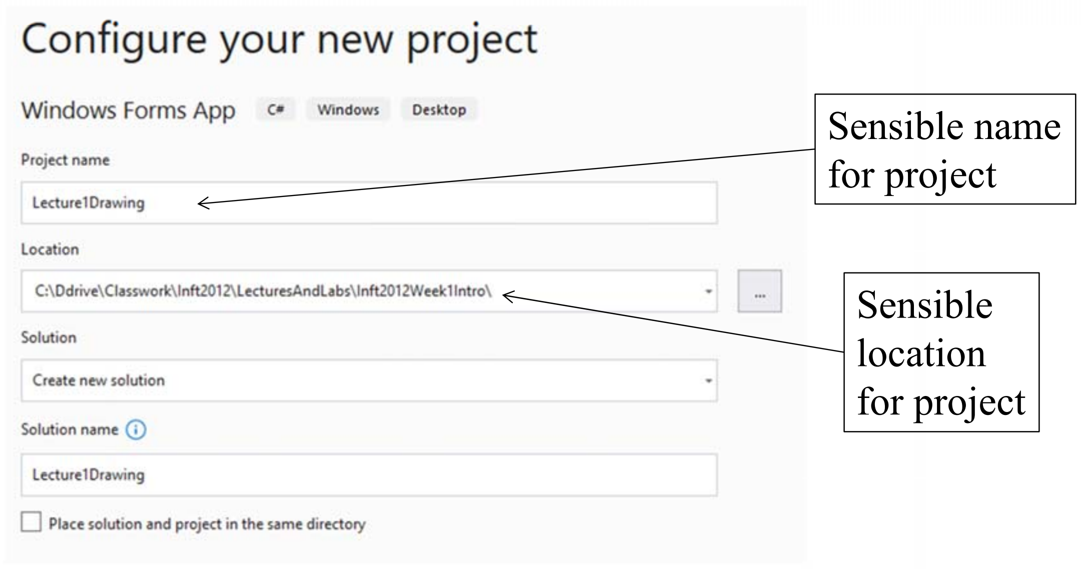
NUnit Test Project (.NET Core)
A project that contains NUnit tests that can run on .NET Core on Windows, Linux and MacOS.
C# Desktop Linux macOS Test Web Windows

Windows Forms App (.NET Framework)
A project for creating an application with a Windows Forms (WinForms) user interface
C# Desktop Windows

WPF App (.NET Framework)
Windows Presentation Foundation client application
C# Desktop Windows XAML

WPF App (.NET Core)
Windows Presentation Foundation client application
C# Desktop Windows

WPF Custom Control Library (.NET Core)
Windows Presentation Foundation custom control library
C# Desktop Library Windows



Configure your new project

Windows Forms App C# Windows Desktop

Project name: Lecture1Drawing

Location: C:\Ddrive\Classwork\Inft2012\LecturesAndLabs\Inft2012Week1Intro\

Solution: Create new solution

Solution name: Lecture1Drawing

☐ Place solution and project in the same directory

Things to do next

- Before you go any further, change the name of the file the form is stored in. Select *Form1.cs* in Solution Explorer, then click in it, then type a more informative name, such as *Drawing.cs*
- If you're asked a question, answer Yes
- This also changes the name of the form itself (a different thing from the file)
- Click on the form. *Drawing* should show near the top of the Properties window, and the A-Z button should be selected, listing the properties in alphabetic order. Find (Name) at the top, and you will see that the value to its right is *Drawing*. Change this to *FrmDrawing*.

Things to do next

- You should also change the form's Text value, which appears in its title bar. In the Properties window, scroll down to the Text property and change it to *Drawing in C#*.
- Every time you start a new project you should change
  - the name of the *.cs* file,
  - the name of the form (starting it with *Frm*), and
  - the text value of the form.
- If you don't remember to do it at the start, you'll probably never remember

Saving your work

- As with any work on the computer, it's important to save your programs regularly
- As with any work on the computer, it's a good idea to do your first save right at the beginning
- Start by clicking the Save All button (a pair of floppy discs) on the standard C# toolbar

Don’t ever use "Save As"!

- As you’ll see if you look where you saved your program, it consists of a large number of files, some in separate folders with names such as *bin* and *obj*
- If you use Save As to save in a different location, you’re relocating just a single file. This will wreck your project. *Don’t do it! Don’t ever use Save As!*
- Whenever you run the program, all files are saved
- If you want to save in the meantime, use the Save All button again
- If you want a copy of the project, wait till you’ve finished and quit, then copy and paste the whole folder with your project in it

Adding controls to the form

- ‘Controls’ are what we call the things we put on the form
- We add a control by
  - dragging it from the toolbox
  - or selecting it in the toolbox then clicking on the form
  - or (if you want a non-standard size) selecting it in the toolbox then dragging its size and location on the form
- In the Toolbox, shrink *All Windows Forms* and expand *Common Controls*, then put a biggish PictureBox and a standard Button on the form

Renaming controls

- Apart from the occasional label, every control we put on a form will be referred to in the program code
- Controls that are referred to by the program must have informative names, names that tell *the programmer* what sort of control they are and what they are used for
- C# doesn't care what they are called, so it gives them sequential names such as *button1* and *pictureBox1*
- We <u>never</u> leave the default names unless we know the program won't have to refer to them
- And we change the names <u>as soon as we've added the controls</u>, to avoid later confusion

Choosing names for controls

- Start with a prefix that tells us the type of the control, eg *Frm* for a form, *Btn* for a button, *Lbl* for a label
- Then add a word or two to describe the purpose of the control, eg
  - *BtnDraw* for a button with Draw on it
  - *PicbxDrawing* for a picture box that we draw in
  - *LblResult* if we were using a label to display a result
- Use 'Pascal case': no spaces, and a capital letter for each word
- Use the properties window to change the names of your controls; also change the Text of your button to *Draw*.

Running the program

- In the standard C# toolbar is a little right-pointing green arrow with “Start” next to it
- When you click this button, C# first saves all the files, then checks that everything in your program makes sense, then runs the program, opening the form on your screen
- Try it now
- The form opens, but nothing happens when you click the button; even so, your program is running
- Close the form; this stops the program

What program?

- What program? All you’ve done is design a form!
- While you were doing that, C# was writing a program to display that form and run it – but that’s all
- To see the program, in Solution Explorer, click the *Drawing.Designer.cs* file; if you don’t see that file, first click the plus next to *Drawing.cs*
- Now, in the code window, click the “+” next to “Windows Form Designer generated code”
- This is the code that creates and displays the form; after taking a brief look at it, close its tab along the top of the main window, and it’ll go away

Writing our own program

- Now we’re going to write some code of our own: an event handler for *BtnDraw*
- First go back to the form design (using the “Drawing.cs [Design]” tab at the top of the main working area
- Double-click the button you placed on the form
- This takes you to the code window for your part of the program, and the cursor is between braces (curly brackets) in a section saying “private void BtnDraw_Click(object sender, EventArgs e)”
- This is the start of an event handler for the button, and we’re going to write the rest of it

Event handler for the button

- Enter this code between the braces of the event handler:

```
Graphics graPaper = picbxDrawing.CreateGraphics();
Pen penBlack = new Pen(Color.Black);
graPaper.DrawRectangle(penBlack, 10, 10, 100, 50);
graPaper.DrawRectangle(penBlack, 10, 75, 100, 100);
Pen penBlue = new Pen(Color.Blue, 3);
graPaper.DrawEllipse(penBlue, 10, 10, 100, 50);
SolidBrush brshGreen = new SolidBrush(Color.Green);
graPaper.FillEllipse(brshGreen, 10, 75, 100, 100);
```

- See how Visual Studio indicates syntax errors, and how it helps you to fix them
- Run the program
- It will be explained in detail next week

What to do before next week

- Get a copy of Visual C# on your computer
  – It’s a big download; be prepared for that
- You can install Visual Studio Community Edition, which is free, and fine for this course
- Open and examine Lecture1DemoTickle; you can open it by double-clicking *Lecture1DemoTickle.sln*
- Create and run the Drawing program described in this lecture – and add some more drawing of your own

Other operating systems

- Microsoft makes Visual Studio (including C#) for Windows and Mac operating systems
- If you want to run Microsoft C# on a Linux Machine, you can run Mono, which is Open Source Software (which is free) that includes a version of C#
- However, the Mono version of C# does not provide the same level of Integrated Development Environment as in Visual Studio
- Also, it is generally some way behind the current Microsoft version, so the two are not generally interchangeable

That’s the end of the lecture . . .

- This week there are no lab classes
- This is not an opportunity to forget this course until next week: it’s an important opportunity to get the software on your computer and start programming

- You are strongly advised to bring a USB drive (or a laptop) to the labs, so that you can transfer your programming work between the lab, other computers at uni, and home