IE531: Algorithms for Data Analytics
Spring, 2018
Lp_Solve & Lp_Solve API Installation Instructions
©Prof. R.S. Sreenivas

The latest version of Visual Studio and XCode have a different look-and-feel in terms of window-layouts etc. That said, you should be able to work through the installation without much hand-holding. Contact the TA (and then me) if you are unable to proceed. Part of the computing experience is developing a stomach for software-installation ☺.

1 Mac Xcode

The illustrations you see in these notes are for *Xcode 3.2.6*. There is a newer version, *Xcode 4.4.1*[1], which is similar in many ways to the older version, but keep in mind the executables are placed in folders with differing names etc. These instructions should work for the latest version too. Come see me if you run into issues.

1. Download lp_solve_5.5.2.0_source.tar.gz from here[2]. Unzip the file to get a tar file, move the tar file to a spot where you want. Untar/unzip it by double-clicking the tar file. You should have a folder called lp_solve_5.5 alongside the tar file.

2. Open the *Terminal* console. On the window that opens up, change directory (cd < *pathname* >) to the directory where folder lp_solve_5.5 is located. Then change directory to lp_solve and type "sh ccc.osx" (cf. figure 1).

3. If you get a bunch of error messages on your screen[3] that says

    cc1: error: unrecognized command line option "-Wno-long-double"

    you have to edit the file ccc.osx using an editor like TextEdit on you MAC applications directory and change the line that reads

    opts='-idirafter /usr/include/sys -O3 -DINTEGERTIME -Wno-long-double'

    to

    opts='-idirafter /usr/include/sys -O3 -DINTEGERTIME'

---

[1] See Section 3 for some preliminaries that you need to complete for Xcode 4,4.1.

[2] If you run into trouble with missing files etc, you might want to download an earlier version from the parent URL here

[3] This depends on the version of MAC OS you are running.

Figure 1: Screenshot relevant to step 2.

(i.e. delete the -Wno-long-double in the end of line before the quotes). Save the ccc.osx file and type "sh ccc.osx". You might get a few warning messages on the screen, but just ignore it. When everything is done, you should have the executable version of lp_solve in the directory bin/os64. This is the MILP solver lp_solve.

4. Type cd ../lpsolve55 and move to the folder/directory called lpsolve55 and type "sh ccc.osx". Like before, if you get a bunch of error messages on your screen[4] that says

   cc1: error: unrecognized command line option "-Wno-long-double"

   you have to edit the file ccc.osx using an editor like TextEdit on you MAC applications directory and change the line that reads

   opts='-idirafter /usr/include/sys -O3 -DINTEGERTIME -Wno-long-double'

   to

   opts='-idirafter /usr/include/sys -O3 -DINTEGERTIME'

   (i.e. delete the -Wno-long-double in the end of line before the quotes). Save the ccc.osx file and type "sh ccc.osx". You might get a warning message on your screen that says something like – fortify.o has no symbols. Just

---

[4]This depends on the version of MAC OS you are running.

ignore it. The directory/folder ls bin/osx64/ should contain a dynamic library liblpsolve55.dylib that we are not going to use. You should delete it. The only file left in this folder would be liblpsolve55.a, we will use this in *Xcode*.

5. For this step you will check if the API works. Open Xcode and open a new project and replace the dummy-code in main.c with the code in demo.c in the lpsolve55 directory/folder that contained the un-tared files downloaded earlier. Add the linker flag -llpsolve55 in the *Other Linker Flags* filed as shown in figure 2.
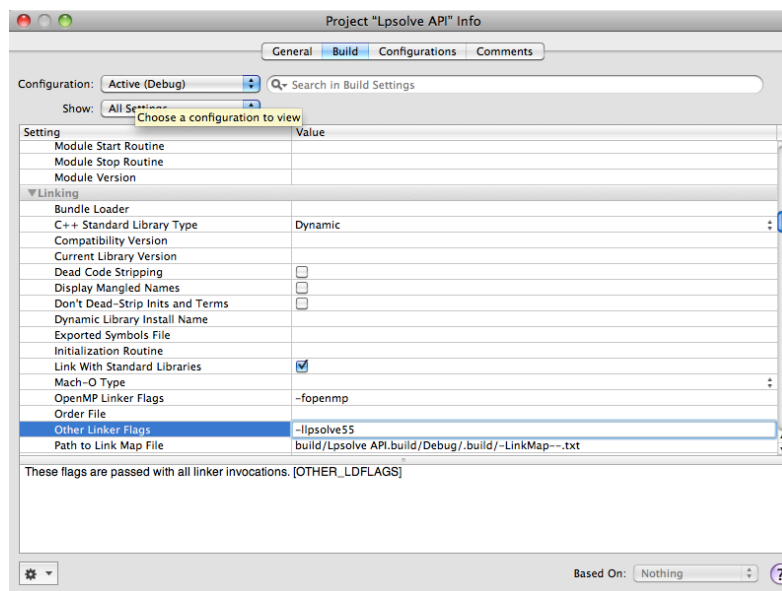
Figure 2: Screenshot relevant to step 5.

6. Fill the *Library Search Paths* with the pathname along the lines of $<$ *blah* $>$/lp_solve_5.5/lpsolve55/bin/osx64. Similarly, fill the User Header Search Paths with the pathname along the lines of $<$ *blah* $>$/lp_solve_5.5 (cf. figure 3). Compile the code.

7. If you ran the compiled code, you should get something along the lines of what is shown in figure 4. You are now ready to use the *Lpsolve API* within your C/C++ code. You will have to press the return button a couple of times for the process to proceed.

The *lp_solve Reference Guide* can be found here. The lp_solve API reference can be found in one of the menu items on the left-hand-side (cf. figure 5). Now, you can include any of these commands (with appropriate initializations, of course) into your C/C++ code, with this you can incorporate a non-trivial *Mixed, Integer Linear Programming* (MILP) solver into your code with ease!
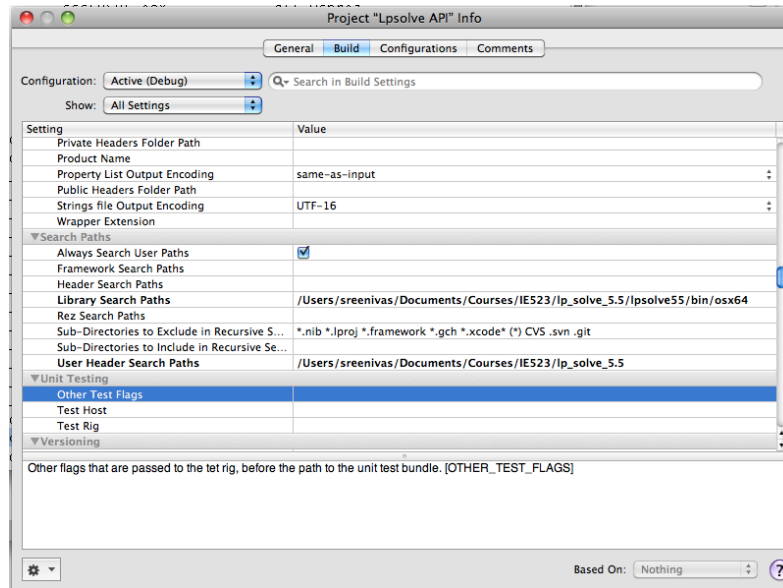
3

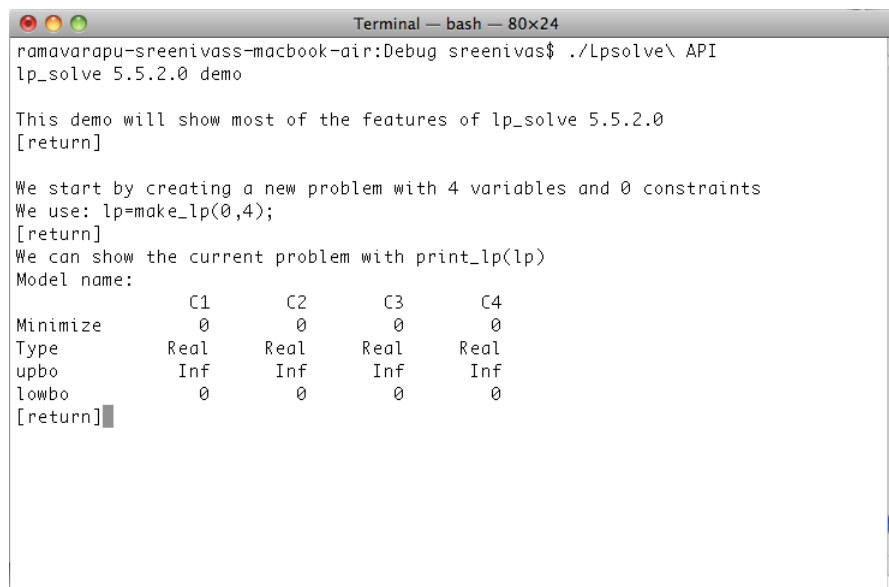Figure 3: Screenshot relevant to step 6.
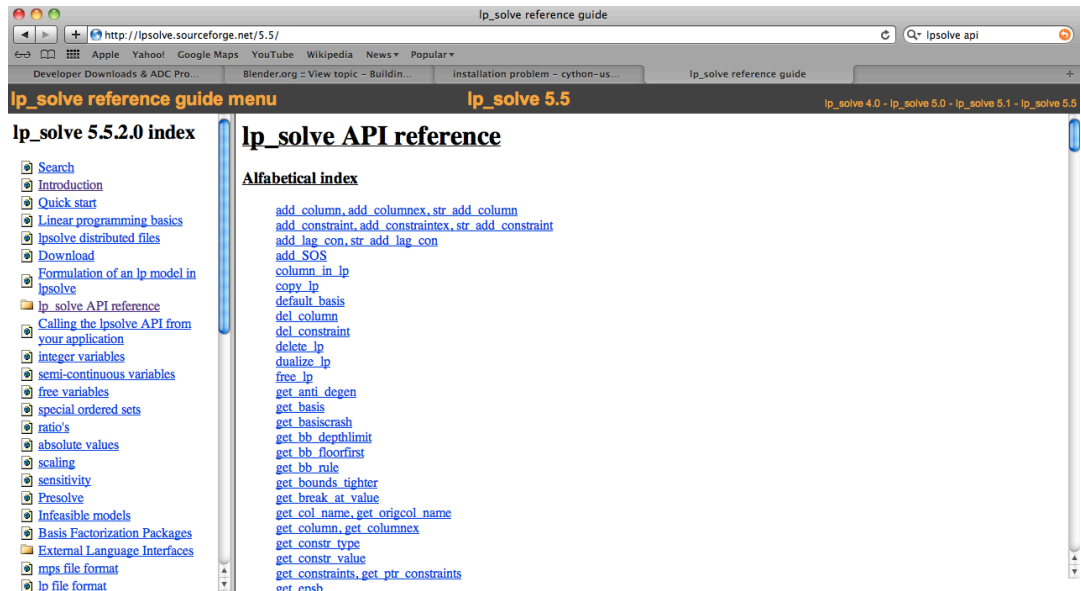

Figure 4: Screenshot relevant to step 7.

Figure 5: *lp_solve Reference Guide* and the *lp_solve API reference* menu item.

2 Visual Studio

1. Download lp_solve_5.5.2.0_source.tar.gz from here. Unzip/Untar downloaded file after moving it to a desired location. I am leaving it in the Downloads directory of my machine. When you are done you will have folder/directory called lp_solve_5.5, double click it, and then double-click the lp_solve folder/directory (cf. figure 6).

2. Double-click the lp_solve.sln icon. Depending on the version of Visual Studio that you have, you might get a conversion-dialog window. Just step through it normally ? this coverts the lp_solve.sln folder to compatible with the version of Visual Studio that you have in your computer. Build the solution. When I ran it on my computer I got a

   LINK : fatal error LNK1104: cannot open file 'Win32/Debug/lp_solve.map'

   error. I just ran the build again (I guess it is looking for file called lp_solve.map; and it is created when the first build is done or something) and everything worked fine (cf. figure 7)

3. You should have the *MILP solver* lp_solve.exe in the lp_solve_5.5/lp_solve/Win32/Debug folder. You can test it if you wish. Create a file called test.txt in Notepad with the following lines:
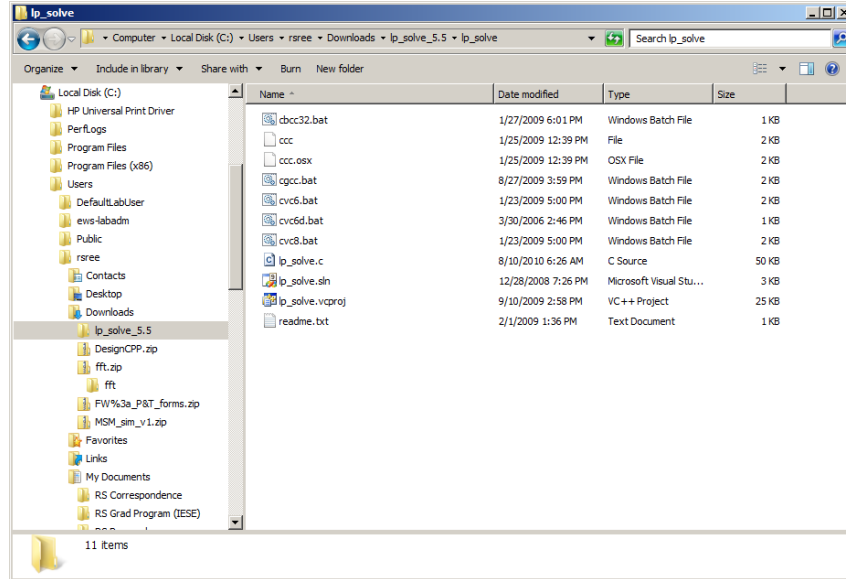
   max: 7x1+5x2;
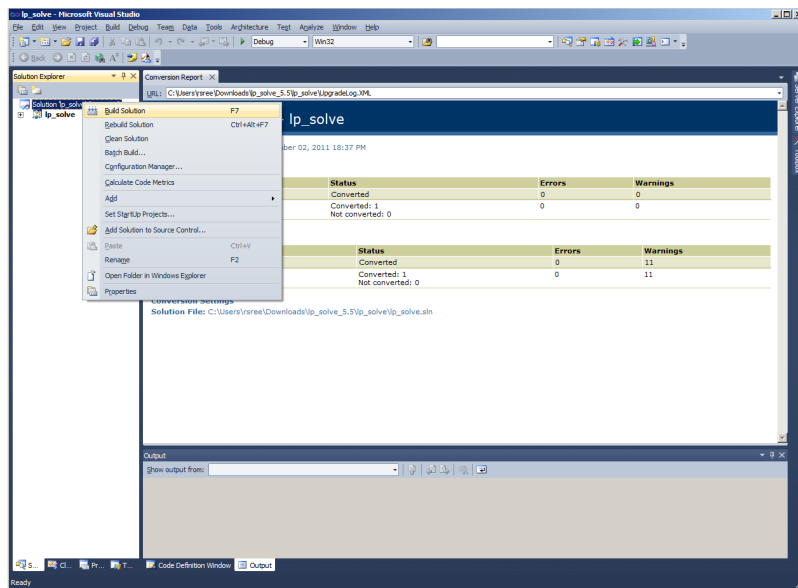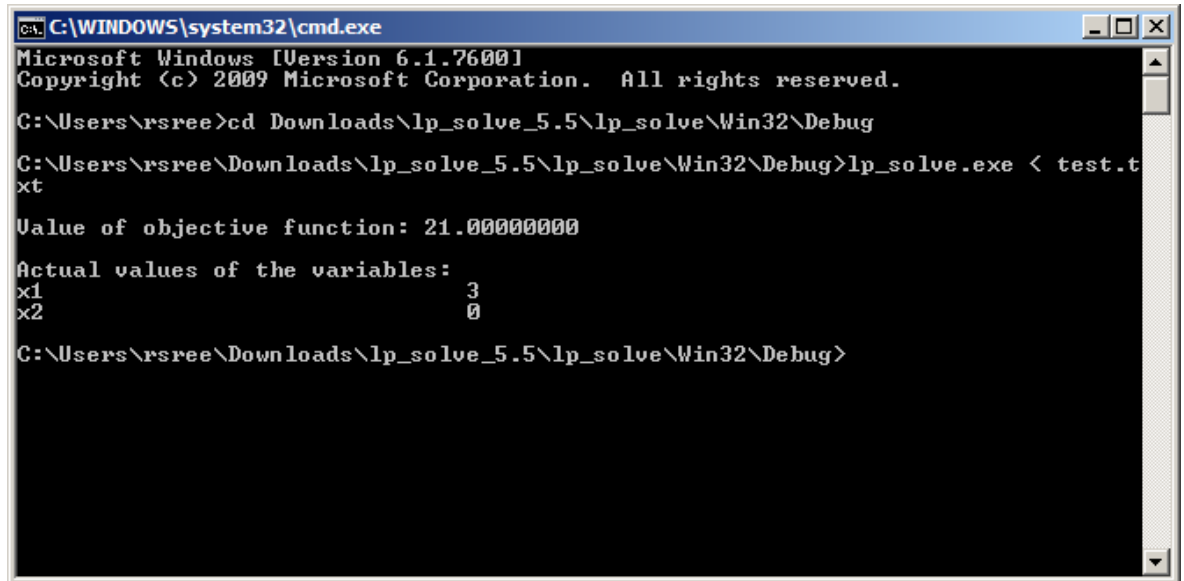   x1 + 2*x2 <= 6;

5

Figure 6: Screenshot relevant to step 1.

Figure 7: Screenshot relevant to step 2.

4*x1 + 3*x2 <= 12;

Save it in the the lp_solve_5.5/lp_solve/Win32/Debug folder. Using the cmd window move to the the lp_solve_5.5/lp_solve/Win32/Debug and type lp_solve.exe < test.txt on the command line. You get the optimal value to be 21 with x1 = 3 and x2 = 0 (cf. figure 8)

Figure 8: Screenshot relevant to step 3.

4. Now, you are going to create the library file for the lp_solve API (which is very similar to what you did not NEWMAT). For this part you will move the lp_solve_5.5/lpsolve55 directory and double click the lib.sln icon (cf. figure 9)

5. Go through any conversion steps (if any) and build the solution ?lib? (cf. figure 10).

6. Now, let us check if the library files work. Go to the lp_solve_5.5/demo directory/folder and double-click the demolib.sln icon and go through the conversion steps (if any). Try building it. You might get an error that says

LINK : fatal error LNK1104: cannot open file 'liblpsolve55d.lib.

If you do then you have to make some changes via the Properties menu item (cf. figure 11)

7. Navigate to the linker − > input − > Additional Dependencies menu item (cf. figure 12)).
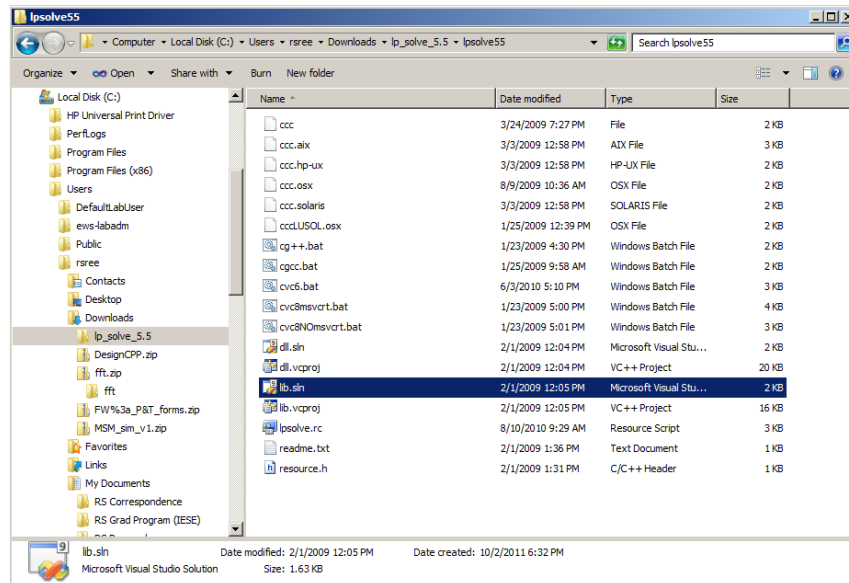
7

Figure 9: Screenshot relevant to step 4.
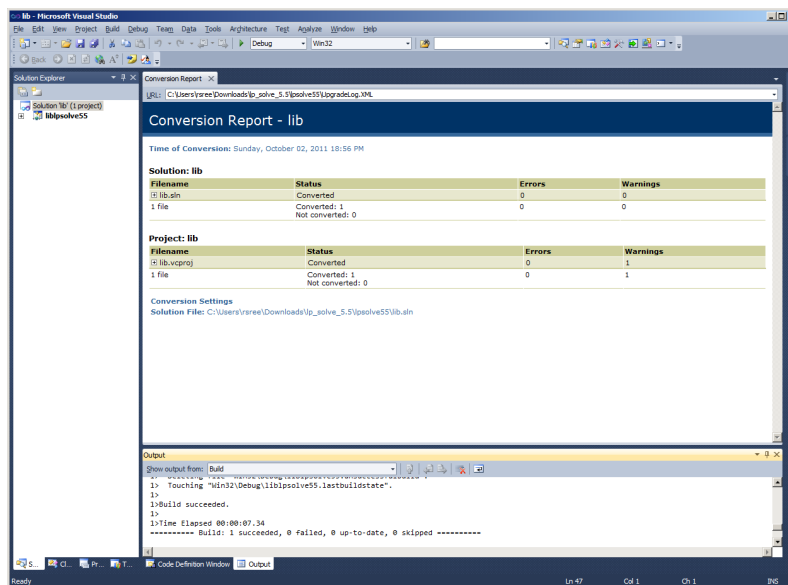

Figure 10: Screenshot relevant to step 5.

8
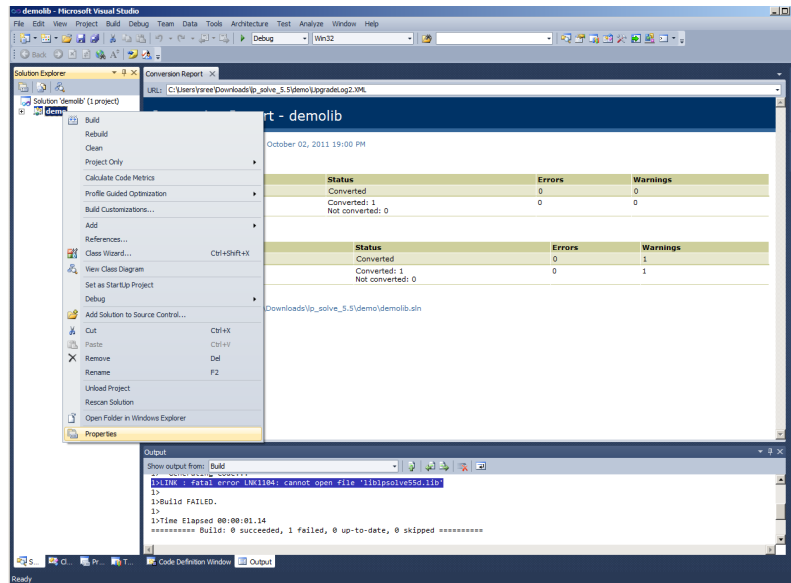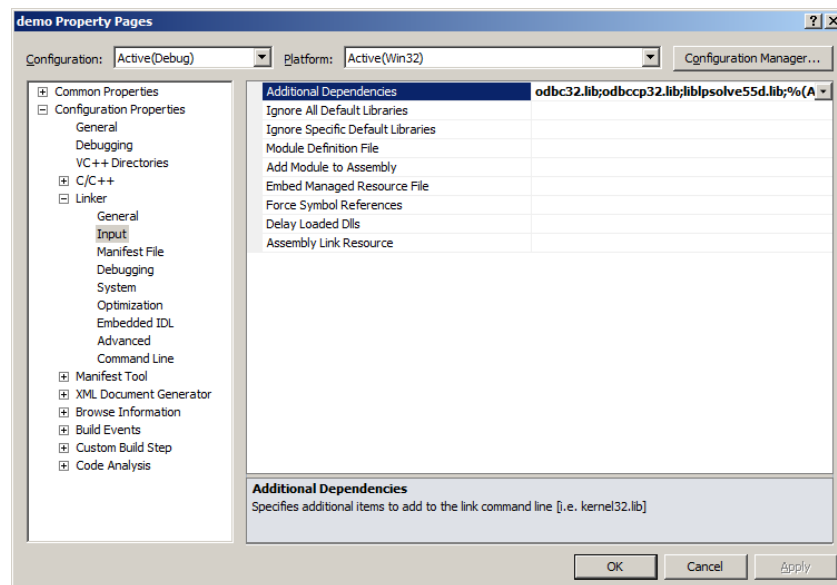
Figure 11: Screenshot relevant to step 6.

Figure 12: Screenshot relevant to step 7.

8. Click the <Edit?> menu item. Change the liblpsolve55d.lib to the liblp-solve55.lib (i.e. remove the "d" in the name of the library) as shown in figure 13. Click OK and and you should get something like what is shown in figure 14.
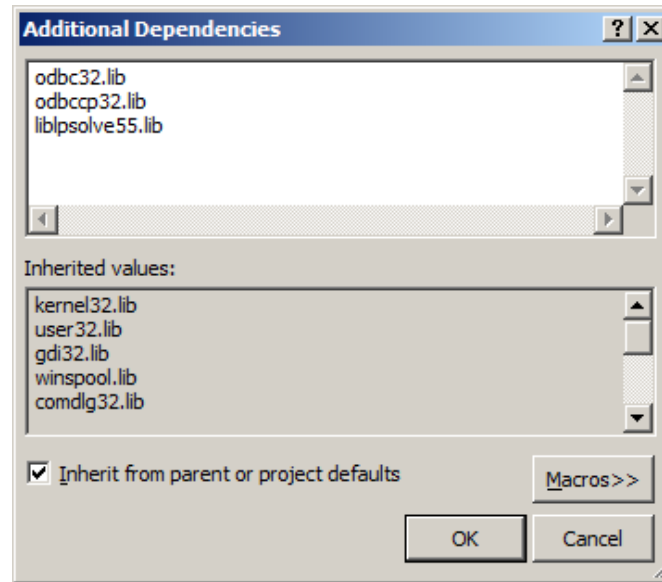

Figure 13: Screenshot relevant to step 8.

9. Now, go to the linker − > General − > Additional Library Directories menu item. Change (using the Edit... menu, as before) the pathname ../lp-solve55/bin/win32 to ../lpsolve55/bin/win32/Debug and click OK.

10. Rebuild. Move to the directory ../demo/Win32/Debug and run demo.exe. You should get something like what is shown in figure 15, and you are ready to use the lp_solve API.

3   Some Preliminaries for XCode4.4.1

XCode 4.4.1 is slightly different from the previous versions of XCode. You can download the latest version of XCode using the *Mac App Store*.

1. Open XCode after it has completed installation. Open the "Preferences" menu item. Select "Downloads" (cf. figure 16) and select the "Command Line Tools" and install it (cf. figure 17).

2. The following steps will locate the executable-files at a spot that you can access. Otherwise, it will place it at the folder where Xcode is installed,
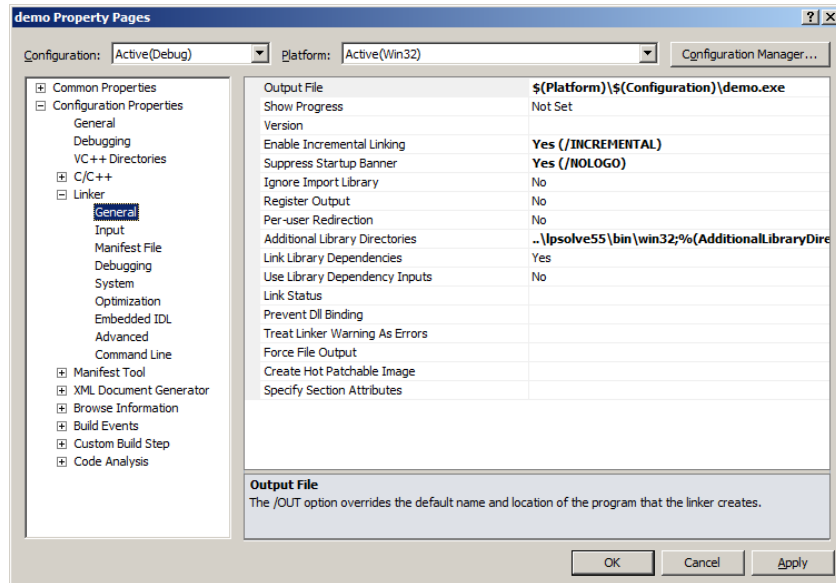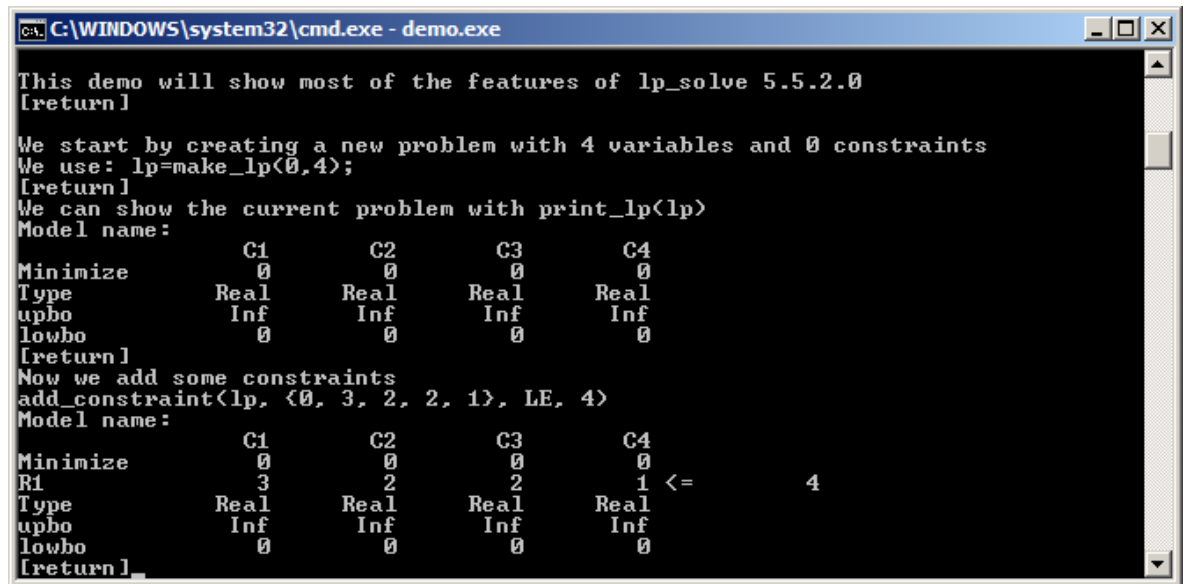
Figure 14: Screenshot relevant to step 8.

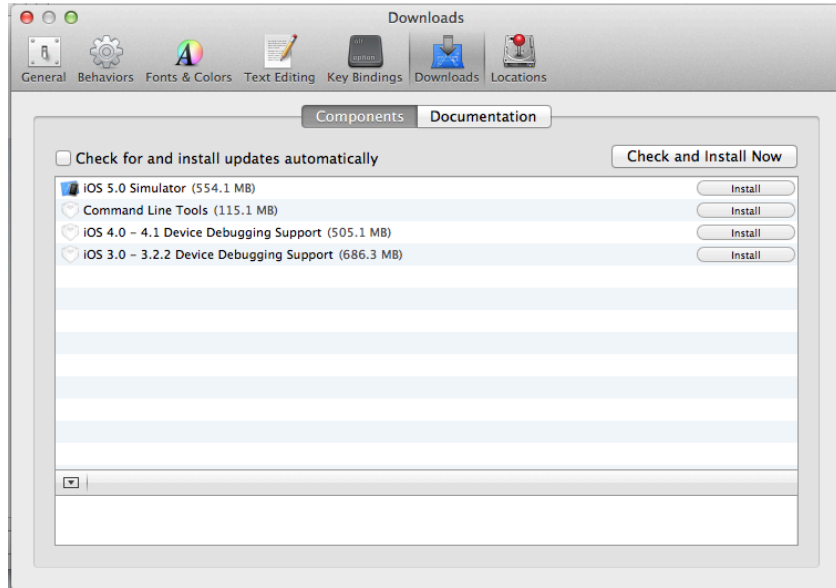Figure 15: Screenshot relevant to step 10.

Figure 16: Screenshot relevant to the first part of step 1.
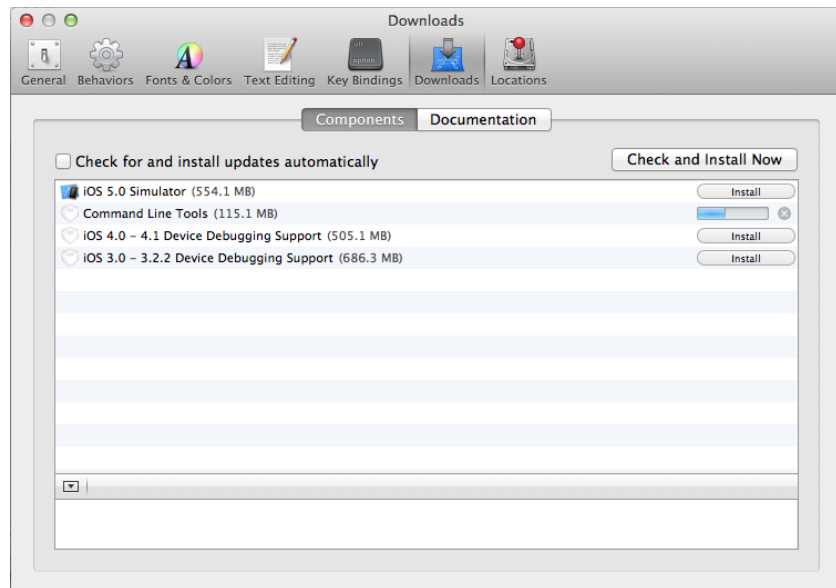

Figure 17: Screenshot relevant to the second part of step 1.

which can be a pain to get to later on. On the "Preferences" menu, pick the "Locations" menu-item and for the "Derived Data" item, select "Relative" (cf. figure 18). The executable file will be placed in

```
<project folder>/Build/Products/Debug
```
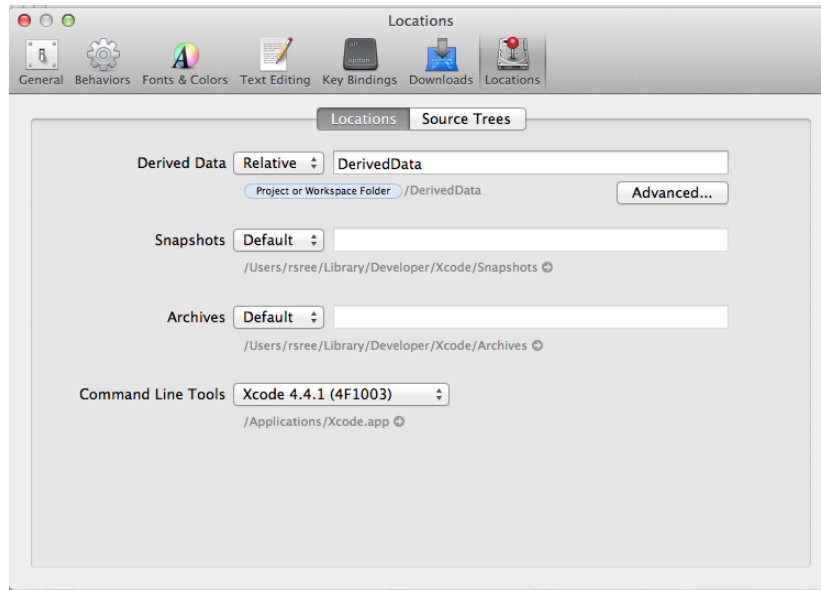
directory.

Figure 18: Screenshot relevant to the second part of step 2.

3. The "look-and-feel" of the interface where you tell the Compiler about the linker flags, header and library paths and the architecture is slightly different from the earlier versions. I thought I should include the screen shots of these windows just for your convenience (cf. figures 19, 20 and 21).

13

Figure 19: Screenshot relevant to the second part of step 3. This is where the linker flag is set.

Figure 20: Screenshot relevant to the second part of step 3. This is where the header and library paths are set.
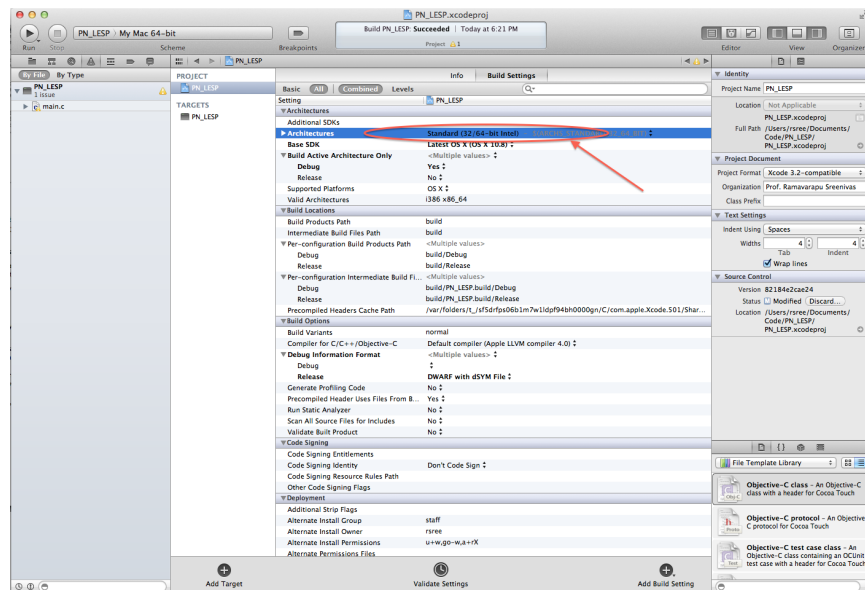
Figure 21: Screenshot relevant to the second part of step 3. This is where the architecture flag is set.