

# Week 02 Laboratory Sample Solutions

## Objectives

- Understanding use of UNIX pipelines
- Understanding use of UNIX filters (sed, sort, uniq, cut, tr)

## Preparation

Before the lab you should re-read the relevant lecture slides and their accompanying examples.

## Getting Started

Set up for the lab by creating a new directory called `lab02` and changing to this directory.

```
$ mkdir lab02
$ cd lab02
```

There are some provided files for this lab which you can fetch with this command:

```
$ 2041 fetch lab02
```

If you're not working at CSE, you can download the provided files as a [zip file](#) or a [tar file](#).

### EXERCISE:

## Sorting UNSW Enrolments

There is a template file named `sorting_enrolments_answers.txt` which you must use to enter the answers for this exercise.

The autotest scripts depend on the format of `sorting_enrolments_answers.txt` so just add your answers don't otherwise change the file.

The file `enrolments.psv` contains a list of CSE enrolments.

The file `enrolments.psv` has 9 columns of data (columns are pipe separated):

1. UNSW Course Code
2. UNSW zID
3. Name
4. UNSW Program
5. UNSW Plan
6. WAM
7. UNSW Session
8. Birthdate
9. Sex

Each row of data represents one enrolment.

1. Write the `sort` and the `head` or `tail` commands needed to print the enrolment for the student with the lowest *zID*.

### HINT:

It should print:

```
COMP9311|5200002|Chen, Jason Yi Kenny  
|8543|COMPA1|077.185|22T2|19971102|M
```

Note that this is one line.

The line is probably too long to display without wrapping in most browsers.

The same is true for the other parts of this activity.

For a better view, copy the expected output into a text editor.

#### ANSWER:

Sample answer:

```
$ sort -t'|' -k2,2 enrolments.psv | head -1
```

Approach:

Sort using '|' as the field delimiter.

Sort by zIDs(col 2).

Grab the lowest (first) zID ( head ).

Note: -kx,y means start sorting on xth column and stop sorting end of yth column

As always autotests are available

```
$ 2041 autotest sorting_enrolments Q1
```

2. Write the `sort` and the `head` or `tail` commands needed to print the first 100 enrolments ordered first by *Course Code*, then by *zID*.

#### HINT:

It should print:

BINF3010 5200973 Gupta, Yiming  8543 AEROAH 031.007 22T2 20020322 M	
BINF3010 5202121 Cheung Ben, Ali Rohan Edward MTRNAH 026.301 22T2 20011005 M	3764/3 COMPA1
BINF3010 5209091 Islam, Justin Wei Kai  3778/3 COMPBH 088.132 22T2 20011118 M	
BINF3010 5214458 Yao, Andy Terry  3707/1 COMPI1 093.226 22T2 19930219 M	
BINF3010 5215095 Tao, Evan George Zhen  3956/1 CVENAH 040.530 22T2 19940414 M	
BINF3010 5215758 Liu, Ryan  3707/1 COMPA1 098.827 22T2 19980909 M	
BINF3010 5218208 Chen, Edward Adam  3784/2 COMPSS 080.555 22T2 19970903 M	
BINF3010 5222596 Yao, Vincent Xiang  3789/1 COMPFR 047.259 22T2 20010804 M	
BINF3010 5227223 Pan, Callum Md Jacob  3707/1 COMPER 051.923 22T2 20041016 M	
BINF3010 5229113 Wu Kenneth, Kevin John  8543 COMPA1 061.847 22T2 19991126 M	
BINF3010 5229808 Lai, Tina Sophia Yi MATHM1 058.312 22T2 20030303 F	2645 COMPA1
BINF3010 5232806 Zhao, Alex Jin Dan  3778/2 COMPA1 069.475 22T2 20041011 M	
BINF3010 5235514 Hao Evan, Oscar Yi  8621 DPENG1 071.537 22T2 19850605 M	
BINF3010 5236229 Kim, Yi Charlotte  3707/2 COMPSS 026.743 22T2 19990827 F	
BINF3010 5236296 Xu, Sam Boyang U 092.089 22T2 19990309 M	3768/5 UNDL-
BINF3010 5241990 Lu, Patrick Daniel William Marcus  3784/1 COMPSS 065.526 22T2 19981211 M	
BINF3010 5243079 Xiao, Zheng Oliver  3778/2 BINFAH 075.843 22T2 19961219 M	
BINF3010 5244610 Yin Hao, Gary Ziheng  3767/2 CODEA1 051.530 22T2 20010619 M	
BINF3010 5244987 Leung Jonathan, Jiawei Jin MECHAH 072.821 22T2 20040128 M	8750/1 COMPI1
BINF3010 5246640 Liao, Ryan  3673/3 SOLABH 086.638 22T2 20000308 M	
BINF3010 5253879 Chen, Jack  3778/2 COMPA1 066.779 22T2 19940316 M	
BINF3010 5257078 Ren, Christina Jessica COMPSS 077.351 22T2 19950721 F	3674/4 COMPAS
BINF3010 5260254 He, Yu George Joseph  7004/1 COMPCS 064.663 22T2 19920101 M	
BINF3010 5261549 Nguyen, Jiaqi  3785/1 BIOMDS 072.215 22T2 19781112 F	
BINF3010 5269596 Jin, Xinyi  7021/1 GMATBR 058.274 22T2 20020121 F	
BINF3010 5272952 Chen, Ray  8338 AEROAH 083.855 22T2 20010831 M	
BINF3010 5276189 Zhao Yi, Benjamin  7003/1 CEICAH 023.157 22T2 20030824 M	
BINF3010 5277078 Zhu, Xinyu Jiayi Jiayi Catherine  1550 DPENM1 048.533 22T2 20001023 F	
BINF3010 5286087 Nguyen, Xin Michelle Olivia  3778/3 COMPA1 085.168 22T2 19980730 F	
BINF3010 5288172 Huang, Jiaqi  3768/3 DPBSA1 048.122 22T2 19970821 F	
BINF3010 5289225 Zhang, Nicholas Ali Ho  3707/1 COMPA1 044.509 22T2 20040410 M	
BINF3010 5289226 Liu, Daniel Ran  8543 COMPA1 083.666 22T2 20030922 M	
BINF3010 5292898 Feng, Jonathan Tian  3762/2 HDATAS 055.582 22T2 20010511 M	
BINF3010 5295853 Wang, Andrew Peter Kevin  3707/1 COMPA1 082.708 22T2 20010504 M	
BINF9010 5217385 Wong, Peter Jie  8543 MTRNAH 066.824 22T2 19980603 M	
BINF9010 5256553 Zhang, Elizabeth Sophie	

3707/1 COMPA1 084.707 22T2 19970922 F	
BINF9010 5260608 Yang, Peter Jackson	
1650 SENGAH 041.084 22T2 19980430 M	
BINF9010 5263488 Feng, Md Oscar	
3673/3 COMPDS 058.408 22T2 20000831 M	
BINF9010 5266191 Lim, Hao Simon	
3778/1 COMPCS 039.280 22T2 19991008 M	
BIOM1010 5200130 Huang, Michael Max	
3778/1 COMPA1 088.662 22T2 20021208 M	
BIOM1010 5200367 Zhou, Adam Edwin	
3707/1 COMPA1 090.978 22T2 19781112 M	
BIOM1010 5201672 Liang, Raymond Robert Sebastian	3778/3 UNDL-
U 023.238 22T2 19991224 M	
BIOM1010 5201771 Le, John Ken Stanley	
3959/1 COMPCS 074.570 22T2 20040118 M	
BIOM1010 5202753 Lin, Steven Marcus Liam	3778/3 UNDL-
U 071.970 22T2 19990409 M	
BIOM1010 5203125 Shi, Michael Alexander	
8543 COMPER 064.241 22T2 19950813 M	
BIOM1010 5205285 Chan, Chloe Jiaqi Yue	
7004/1 COMPA1 055.497 22T2 19951127 F	
BIOM1010 5205500 Chen, David Patrick	
8543 COMPA1 088.514 22T2 19920423 M	
BIOM1010 5206339 Guo Hugo, Joshua Yu	
8543 COMPAS 038.556 22T2 20020831 M	
BIOM1010 5206835 Singh Peter, Haoran	
3791/4 SENGAH 030.967 22T2 20030414 M	
BIOM1010 5207199 Wang, Hannah Jennifer Xinran	3778/3 COMPA1
CVENAH 075.810 22T2 19960723 F	
BIOM1010 5209023 Lee, Nicholas Leon	
3707/2 COMPCS 064.980 22T2 20020403 M	
BIOM1010 5211169 Huang, Benjamin Zihao Ryan	
3133/4 PHYSL1 057.434 22T2 19930812 M	
BIOM1010 5212059 Li, Grace Melanie	
8543 COMPA1 086.331 22T2 19840103 F	
BIOM1010 5212421 Liu Kenneth, Nicholas Isaac	
3707/3 COMPCS 088.638 22T2 19981116 M	
BIOM1010 5213416 Jain, Callum George Chen	
3673/1 BIOMAR 070.375 22T2 19970625 M	
BIOM1010 5213892 Chen, Callum Rohan Steven	
7004/1 DPENX1 075.902 22T2 19990809 M	
BIOM1010 5215910 Nguyen, Matthew Brendan Zihao Gary	
3778/1 COMPA1 060.684 22T2 20020509 M	
BIOM1010 5216925 Wu, Alex Calvin	
3778/3 COMPA1 072.298 22T2 19980114 M	
BIOM1010 5217306 Liu, Ethan	
8543 COMPA1 092.851 22T2 19740208 M	
BIOM1010 5218400 Chan, Vanessa Xinyi	
3959/1 BINFAH 079.405 22T2 20011001 F	
BIOM1010 5218568 Lee Cameron Xavier, Luke John Christian	
8543 MTRNAH 062.543 22T2 19960313 M	
BIOM1010 5220943 Ahmed, Adam Callum	
1710 COMPAH 050.937 22T2 20030721 M	
BIOM1010 5221389 Jiang, Jing	3707/1 UNDL-
U 081.109 22T2 20000923 F	
BIOM1010 5222122 Zhang, Henry Ray	
3707/2 SENGAH 093.261 22T2 19970730 M	
BIOM1010 5223265 Huang, Olivia Charlotte Catherine	
8543 COMPSS 045.613 22T2 20020217 F	
BIOM1010 5226153 Zou, Darren Jonathan Brandon	
3707/3 COMPCS 086.684 22T2 20030131 M	
BIOM1010 5226489 Qin, Andrew Oliver	
3781/1 COMPAS 076.178 22T2 20011004 M	
BIOM1010 5226606 Khan, Adrian Frank Luke	
7004/1 COMPY1 083.017 22T2 19970709 M	
BIOM1010 5228112 Lin, James	
3707/1 MTRNAH 072.157 22T2 19990603 M	
BIOM1010 5229388 Yu, Hao Jacky	
3784/1 COMPA1 075.401 22T2 19971231 M	
BIOM1010 5229455 Ang, Jasper Martin	
7003/1 BIOMAR 084.525 22T2 20010407 M	
BIOM1010 5231050 Huang, Michael	3778/3 COMPA1

BIOM1010 5231250 Liu, Catherine	3783/2 COMPAS
COMPSC 072.609 22T2 20010306 F	
BIOM1010 5231281 Bai, Zihao Yiming Josh	
1710 CVENAH 065.158 22T2 19710321 M	
BIOM1010 5231659 Zhang, Emily Xin Chelsea	
8543 COMPA1 088.845 22T2 19961028 F	
BIOM1010 5231674 Nguyen, Samuel Calvin	
3707/1 COMPSC 077.377 22T2 19980324 M	
BIOM1010 5232231 Wang, Daniel Tom Jake	
8543 COMPA1 086.036 22T2 19910502 M	
BIOM1010 5233409 Luong, Leon Terry	8750/1 BIOCC1
COMPA1 077.152 22T2 19980330 M	
BIOM1010 5233731 Yang, Nicholas Paul Ray Caleb	
3778/3 COMPSC 027.115 22T2 19980103 M	
BIOM1010 5233934 Li, Henry Yuhao	
1650 MECHAH 099.589 22T2 19950222 M	
BIOM1010 5236838 Cao, Victor Joel Alexander	3134/1 FINSR1
MATHU1 026.051 22T2 19770807 M	
BIOM1010 5240534 Chen, Samuel Albert	
3707/1 COMPA1 025.376 22T2 20000325 M	
BIOM1010 5241683 Chen, Aaron	
3778/2 COMPSC 071.956 22T2 19900317 M	
BIOM1010 5244497 Chan, Hao	
3707/4 MTRNAH 022.076 22T2 19990607 M	
BIOM1010 5246074 Song Gabriel, Harry	3761/1 COMPA1
ELECAH 072.462 22T2 19891122 M	
BIOM1010 5246886 Yuan Ray, Harrison Haoran	
3767/5 COMPIS 085.739 22T2 20020129 M	
BIOM1010 5247018 Lin, Will Richard	
3707/2 COMPSC 066.938 22T2 20030321 M	
BIOM1010 5247854 Guan, Paul Brian	
8543 DPSTJ1 059.628 22T2 19950719 M	
BIOM1010 5249696 Huang, Christina Christina	
3778/1 COMPAS 067.975 22T2 19951001 F	
BIOM1010 5249802 Sun, Darren	3785/4 COMPA1
MTRNAH 039.380 22T2 20040524 M	
BIOM1010 5250963 Chan, Jing Rebecca	
8543 MTRNAH 080.128 22T2 20020727 F	
BIOM1010 5252700 Phan, Martin Zheng Owen Jacky Haoyu	3768/1 COMPAS
COMPDS 080.498 22T2 19760216 M	
BIOM1010 5253909 Chen, Jonathan Derek	
3778/2 COMPA1 055.397 22T2 20001228 M	
BIOM1010 5255656 Huang, Yi Zihao James Edward	
8543 COMPA1 050.187 22T2 19970813 M	
BIOM1010 5256426 Ma, Cameron Tong	3502/3 COMPA1
MARKA1 098.696 22T2 20010708 M	
BIOM1010 5259634 Wang, James Robert Jackson	
8543 COMPA1 036.607 22T2 19970503 M	
BIOM1010 5260593 Wang Zihao, Kevin Jun	
3707/1 COMPA1 026.678 22T2 20020830 M	
BIOM1010 5260847 Mo, Feng Liam	
3674/1 COMPD1 045.351 22T2 19991021 M	
BIOM1010 5261112 Lin, Vincent Aryan	
3778/3 COMPSC 060.743 22T2 19940703 M	
BIOM1010 5261220 Dai, Vanessa	3781/4 BIOMDS
MATSF1 069.417 22T2 20040628 F	
BIOM1010 5261757 Wang, Christine Olivia	
3707/1 COMPA1 083.527 22T2 19960911 F	

ANSWER:

Sample answer:

```
$ sort -t'|' -k1,2 enrolments.psv | head -100
```

Approach:

- Sort using '|' as the field delimiter.
- Sort by course codes(col 1) and then zIDs(col 2).
- Grab the lowest (first) enrolments ( head ).

Note: `-kx,y` means start sorting on `x`th column and stop sorting end of `y`th column

As always autotests are available

```
$ 2041 autotest sorting_enrolments Q2
```

3. Write the `sort` and the `head` or `tail` commands needed to print the first 50 enrolments ordered first by *Birthdate*, then by *Course Code*, then by *Zid*.

HINT:

It should print:



COMP2511|5257575|Sun Mia, Yue Jiaqi  
|3778/2|COMPAS|087.762|22T2|19590220|F  
COMP9154|5283647|Zhou, Maria Kelly Michelle  
|8543|COMPA1|060.834|22T2|19590220|F  
COMP9313|5283647|Zhou, Maria Kelly Michelle  
|8543|COMPA1|060.834|22T2|19590220|F  
COMP1511|5232487|Pham, Jie |3956/2|COMPA1  
ECONE1|062.977|22T2|19590630|M  
COMP1531|5287092|He, Edward Jeffrey  
|8543|SENGAH|073.267|22T2|19590630|M  
COMP2511|5203298|Chan, Melanie Emily Sanjana  
|1650|MTRNAH|074.360|22T2|19590630|F  
COMP6443|5232487|Pham, Jie |3956/2|COMPA1  
ECONE1|062.977|22T2|19590630|M  
COMP6447|5247615|Lam, Richard Max Albert Raymond  
|3959/3|MTRNAH|052.006|22T2|19590630|M  
COMP9417|5239634|Cheng, Xinran Olivia  
|3523/2|CVENIT|071.625|22T2|19590630|F  
COMP1521|5212361|Li, Tim Ziheng Chen |3781/1|COMPA1  
FINSAL|069.945|22T2|19610531|M  
BIOM4952|5236800|Wang, William  
|8543|COMPA1|065.474|22T2|19620111|M  
COMP3121|5263831|Luong, Jimmy  
|3674/4|COMPD1|090.143|22T2|19620111|M  
COMP3331|5248093|Ji, Chris Xiang Danny Haoyu Derek |3959/1|COMPA1  
MTRNAH|065.105|22T2|19620111|M  
COMP9313|5236800|Wang, William  
|8543|COMPA1|065.474|22T2|19620111|M  
COMP9417|5263831|Luong, Jimmy  
|3674/4|COMPD1|090.143|22T2|19620111|M  
COMP9444|5236800|Wang, William  
|8543|COMPA1|065.474|22T2|19620111|M  
COMP9900|5236800|Wang, William  
|8543|COMPA1|065.474|22T2|19620111|M  
COMP9900|5263831|Luong, Jimmy  
|3674/4|COMPD1|090.143|22T2|19620111|M  
COMP1010|5248050|Su Hao, Alexander Lawrence Jay Andrew  
|3784/2|SENGAH|034.557|22T2|19630823|M  
COMP2511|5235981|Guo, Natalie Anna  
|3707/1|ELECAH|058.717|22T2|19630823|F  
COMP3151|5249992|Wang, Anthony  
|3707/2|COMPCS|096.538|22T2|19630823|M  
COMP9313|5249992|Wang, Anthony  
|3707/2|COMPCS|096.538|22T2|19630823|M  
DPGE1002|5249992|Wang, Anthony  
|3707/2|COMPCS|096.538|22T2|19630823|M  
COMP1511|5213800|Zhang, James Ran  
|3778/3|COMPA1|061.418|22T2|19650406|M  
COMP1531|5262100|Liang, Kyle Alex Martin Gordon |3785/3|UNDL-  
U|055.432|22T2|19650406|M  
COMP2521|5296049|Cao, Anna Karen  
|3707/1|MECHAH|074.934|22T2|19650406|F  
COMP3900|5262100|Liang, Kyle Alex Martin Gordon |3785/3|UNDL-  
U|055.432|22T2|19650406|M  
COMP3900|5298425|Chowdhury Hugo, Feng Johnny Justin |3761/1|ENGGAH  
PHYSL1|052.397|22T2|19650406|M  
COMP9313|5213800|Zhang, James Ran  
|3778/3|COMPA1|061.418|22T2|19650406|M  
COMP9313|5296049|Cao, Anna Karen  
|3707/1|MECHAH|074.934|22T2|19650406|F  
COMP9414|5298425|Chowdhury Hugo, Feng Johnny Justin |3761/1|ENGGAH  
PHYSL1|052.397|22T2|19650406|M  
COMP9417|5213800|Zhang, James Ran  
|3778/3|COMPA1|061.418|22T2|19650406|M  
COMP9417|5298425|Chowdhury Hugo, Feng Johnny Justin |3761/1|ENGGAH  
PHYSL1|052.397|22T2|19650406|M  
DPGE1002|5296049|Cao, Anna Karen  
|3707/1|MECHAH|074.934|22T2|19650406|F  
COMP1511|5282378|Wang, James Sunny  
|8543|COMPA1|055.025|22T2|19661218|M  
COMP9323|5282378|Wang, James Sunny

```
|8543|COMP A1|055.025|22T2|19661218|M
COMP9417|5282378|Wang, James Sunny
|8543|COMP A1|055.025|22T2|19661218|M
COMP9444|5282378|Wang, James Sunny
|8543|COMP A1|055.025|22T2|19661218|M
DPBS1180|5220820|Wang, Matthew Ricky
|8543|COMP A1|051.538|22T2|19661218|M
DPST1013|5220820|Wang, Matthew Ricky
|8543|COMP A1|051.538|22T2|19661218|M
DPST1014|5220820|Wang, Matthew Ricky
|8543|COMP A1|051.538|22T2|19661218|M
COMP1531|5262108|Zhu, Cameron Joseph
|3784/1|COMPSS|060.285|22T2|19690326|M
COMP1511|5202456|Wu Charles, Eric Zac
|3778/3|COMP A1|067.109|22T2|19691224|M
COMP1511|5214496|Li, Daniel William
|8543|COMP A1|027.498|22T2|19691224|M
COMP2511|5227015|Zhang, Andrew Dylan
|8543|COMP A1|037.293|22T2|19700105|M
COMP6443|5272909|Yu, Lachlan Bill
|3785/1|COMPAS|032.946|22T2|19700105|M
COMP6771|5218967|Gupta, Andre |3789/4|UNDL-
U|037.091|22T2|19700105|M
COMP6771|5227015|Zhang, Andrew Dylan
|8543|COMP A1|037.293|22T2|19700105|M
COMP9312|5235893|Yu, Yuchen George Peter Xiao |3789/1|COMPI1
MTRNAH|080.390|22T2|19700105|M
COMP9417|5272909|Yu, Lachlan Bill
|3785/1|COMPAS|032.946|22T2|19700105|M
```

ANSWER:

Sample answer:

```
$ sort -t'|' -k8,8 -k1,2 enrolments.psv | head -50
```

Approach:

Sort using '|' as the field delimiter.  
Sort by birthdates (col 8).  
Then sort by course codes(col 1) and then zIDs(col 2).  
Grab the lowest (first) enrolments ( head ).

Note: -kx,y means start sorting on xth column and stop sorting end of yth column

As always autotests are available

```
$ 2041 autotest sorting_enrolments Q3
```

4. Write the sort and the head or tail commands needed to print the first 25 enrolments ordered first by the decimal part of the WAM in descending order, then by zID in ascending order, then by Course Code also in ascending order.

HINT:

It should print:



COMP9417 5205588 Han, Frank Joe Dylan Nikhil	
3781/4 DPENH1 078.999 22T2 19720617 M	
COMP1010 5216228 Hu, Robert Ali	3778/2 COMPA1
MTRNAH 067.999 22T2 19980226 M	
COMP1511 5216228 Hu, Robert Ali	3778/2 COMPA1
MTRNAH 067.999 22T2 19980226 M	
COMP9021 5216228 Hu, Robert Ali	3778/2 COMPA1
MTRNAH 067.999 22T2 19980226 M	
COMP3900 5243221 Wei, Nathan	
3764/2 COMPAH 066.999 22T2 20020412 M	
COMP3900 5277339 Huang, Eric Jeremy	
3778/2 COMPA1 074.999 22T2 20010219 M	
COMP9414 5277339 Huang, Eric Jeremy	
3778/2 COMPA1 074.999 22T2 20010219 M	
ENGG1811 5277339 Huang, Eric Jeremy	
3778/2 COMPA1 074.999 22T2 20010219 M	
COMP1521 5206534 Huang, Peter Han	
3707/1 SENGAH 094.998 22T2 19950807 M	
DPGE1002 5206534 Huang, Peter Han	
3707/1 SENGAH 094.998 22T2 19950807 M	
COMP2521 5216733 Hou, Evan Oliver Zihao	
3707/3 MTRNES 091.998 22T2 20001213 M	
COMP6771 5216733 Hou, Evan Oliver Zihao	
3707/3 MTRNES 091.998 22T2 20001213 M	
COMP9902 5216733 Hou, Evan Oliver Zihao	
3707/3 MTRNES 091.998 22T2 20001213 M	
COMP1511 5219209 Zhu, Xin Sophie Grace Chelsea	
1650 SENGAH 065.998 22T2 20001004 F	
BIOM9001 5229035 Nguyen, Richard	
3707/3 SENGAH 065.998 22T2 20020720 M	
COMP4336 5229035 Nguyen, Richard	
3707/3 SENGAH 065.998 22T2 20020720 M	
COMP9517 5229035 Nguyen, Richard	
3707/3 SENGAH 065.998 22T2 20020720 M	
COMP1511 5238334 Wang, Eric Yifei	
3707/1 COMPA1 056.998 22T2 19911019 M	
COMP3900 5238334 Wang, Eric Yifei	
3707/1 COMPA1 056.998 22T2 19911019 M	
COMP9444 5238334 Wang, Eric Yifei	
3707/1 COMPA1 056.998 22T2 19911019 M	
DPBS1150 5261465 Shao, Nathan Chun	1650 COMPA1
MTRNAH 050.998 22T2 19950709 M	
COMP1511 5266078 Dong, Maria Charlotte Jing	3782/1 COMPI1
MTRNAH 047.998 22T2 20010323 F	
COMP9417 5266078 Dong, Maria Charlotte Jing	3782/1 COMPI1
MTRNAH 047.998 22T2 20010323 F	
COMP9900 5283284 Khan Vivian, Rebecca	
3781/2 MANFBH 031.998 22T2 19970708 F	
COMP2041 5285451 Zhang, Matthew Rory Gabriel	
8543 COMPA1 042.998 22T2 20031208 M	

ANSWER:

Sample answer:

```
$ sort -t'|' -k6.5,6rn -k2,2n -k1,1 enrolments.psv | head -25
```

Approach:

- Sort using '|' as the field delimiter.
- Sort by WAMs (col 6) from the fifth character to end of column 6, numerically and in reverse (to grab the highest decimals).
- Then sort by zIDs (col 2) numerically.
- Then sort by course codes (col 1).
- Grab the lowest (first) enrolments ( head ).

Note: -kx.z,y means start sorting on xth column from the zth letter and stop sorting end of yth column

As always autotests are available

```
$ 2041 autotest sorting_enrolments Q4
```

When you think your program is working, you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest sorting_enrolments
```

When you are finished working on this exercise, you must submit your work by running `give` :

```
$ give cs2041 lab02_sorting_enrolments sorting_enrolments_answers.txt
```

before **Tuesday 14 June 12:00** to obtain the marks for this lab exercise.

SOLUTION:

Sample solution for `sorting_enrolments_answers.txt`

```
This file is automarked.

Do not add extra lines to this file, just add your answers.

For example if your answer to Q0 is: "grep -E Andrew words.txt"
Change the line that starts with
    "Q0 answer:"
to
    "Q0 answer: grep -E Andrew words.txt"
-----

1) Write the sort and the head or tail commands needed to print the enrolment for the student with
the lowest zID.
Q1 answer: sort -t'|' -k2,2 enrolments.psv | head -1

2) Write the sort and the head or tail commands needed to print the first 100 enrolments ordered
first by Course Code, then by zID.
Q2 answer: sort -t'|' -k1,2 enrolments.psv | head -100

3) Write the sort and the head or tail commands needed to print the first 50 enrolments ordered
first by Birthdate, then by Course Code, then by Zid.
Q3 answer: sort -t'|' -k8,8 -k1,2 enrolments.psv | head -50

4) Write the sort and the head or tail commands needed to print the first 25 enrolments ordered
first by the decimal part of the WAM in descending order, then by zID in ascending order, then by
Course Code also in ascending order.
Q4 answer: sort -t'|' -k6.5,6rn -k2,2n -k1,1 enrolments.psv | head -25
```

EXERCISE:

Counting UNSW classes

There is a template file named `counting_classes_answers.txt` which you must use to enter the answers for this exercise.

The autotest scripts depend on the format of `counting_classes_answers.txt` so just add your answers don't otherwise change the file.

The file `classes.tsv` contains a list of CSE classes.

The file `classes.tsv` has 7 columns of data (columns are tab separated):

- 1. UNSW course code
- 2. UNSW class id
- 3. CSE class type
- 4. Number of enrolled students

4. Number of enrolled students
5. Class enrolment cap
6. Class time
7. Class Location

Each row of data represents one class.

1. Write a shell pipeline which will print how many classes there are.

HINT:

It should print:

1078

ANSWER:

Sample answer:

```
$ wc -l < classes.tsv
```

Approach:

As each line in the file represents a class,  
the number of classes is equal to the number of lines in the file.

As always autotests are available

```
$ 2041 autotest counting_classes Q1
```

2. Write a shell pipeline which will print how many different courses have classes.

HINT:

It should print:

61

HINT:

[cut](#) will be useful here.

ANSWER:

Sample answer:

```
$ cut -f1 classes.tsv | sort | uniq | wc -l
```

Approach:

Extract just the course codes ( `cut` ).

Sort them into groups of identical course codes ( `sort` ).

Compress each group to size one, giving one line for each course ( `uniq` ).

Count the number of lines ( `wc` ).

As always autotests are available

```
$ 2041 autotest counting_classes Q2
```

3. Write a shell pipeline which will print the course with the most classes, and how many classes are in this course.

If there are multiple courses with the same number of classes, print the course that is alphabetically first.

HINT:

It should print:

```
76 COMP1511
```

ANSWER:

Sample answer:

```
$ cut -f1 classes.tsv | sort | uniq -c | sort -n | tail -1
```

Approach:

Extract just the course codes ( `cut` ).

Sort them into groups of identical course codes ( `sort` ).

Compress each group to size one and count the size of each group ( `uniq` ).

Sort by the size of each group ( `sort` ).

Grab the largest (last) group ( `tail` ).

As always autotests are available

```
$ 2041 autotest counting_classes Q3
```

4. Write a shell pipeline which will print the two rooms most frequently used by non-LAB CSE classes and how often they are used.

If there are multiple rooms that are used by the same number of non-LAB CSE classes, print order them alphabetically.

HINT:

It should print:

```
311 Online
23 Quad G045
```

ANSWER:

Sample answer:

```
$ grep -Fv 'LAB' classes.tsv | cut -f7 | sort | uniq -c | sort -nr | head -2
```

Approach:

Extract the non-tut classes ( `grep` ).

Extract just the room names ( `cut` ).

Sort them into groups of identical room names ( `sort` ).

Compress each group to size one and count the size of each group ( `uniq` ).

Sort by the size of each group ( `sort` ).

Grab the largest (first) group ( `head` ).

As always autotests are available

```
$ 2041 autotest counting_classes Q4
```

5. Write a shell pipeline which will print the most common day and time in the week for classes to start and how many classes start at that time.

If there are multiple days and times that are used by the same number of classes, print the day and time that is alphabetically first.

HINT:

It should print:

```
44 Wed 16
```

## HINT:

[cut](#)'s `-d` option will be useful here.

## ANSWER:

Sample answer:

```
$ cut -f6 classes.tsv | cut -d'-' -f1 | cut -d':' -f1 | sort | uniq -c | sort -n |  
tail -1
```

Approach:

Extract just the class times ( `cut` ).

Remove the ending time ( `cut` ).

Sort them into groups of identical times ( `sort` ).

Compress each group to size one and count the size of each group ( `uniq` ).

Sort by the size of each group ( `sort` ).

Grab the largest (last) group ( `tail` ).

As always autotests are available

```
$ 2041 autotest counting_classes Q5
```

6. Write a shell pipeline which will print the latest time a class will finish.

## HINT:

It should print:

## ANSWER:

Sample answer:

```
$ cut -f6 classes.tsv | cut -d' ' -f2 | cut -d'-' -f2 | sort -un | tail -1
```

Approach:

TODO.

As always autotests are available

```
$ 2041 autotest counting_classes Q6
```

7. Write a shell pipeline which will print a list of the course codes of COMP courses that run 2 or more classes of the same type starting at the same time on the same day.  
(e.g. three tuts starting Monday at 10:00).

## HINT:

It should print:

```
COMP1010
COMP1511
COMP1521
COMP1531
COMP1911
COMP2041
COMP2511
COMP2521
COMP3331
COMP3900
COMP6443
COMP6447
COMP6452
COMP6771
COMP6843
COMP9044
COMP9311
COMP9312
COMP9313
COMP9319
COMP9331
COMP9417
COMP9727
COMP9900
```

**ANSWER:**

Sample answer:

```
$ grep -F 'COMP' classes.tsv | cut -f1,3,6 | cut -d'-' -f1 | sort | uniq -d | cut -f1 | sort | uniq
```

Approach:

TODO.

As always autotests are available

```
$ 2041 autotest counting_classes Q7
```

When you think your program is working, you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest counting_classes
```

When you are finished working on this exercise, you must submit your work by running `give` :

```
$ give cs2041 lab02_counting_classes counting_classes_answers.txt
```

before **Tuesday 14 June 12:00** to obtain the marks for this lab exercise.

**SOLUTION:**

Sample solution for `counting_classes_answers.txt`



This file is automarked.

Do not add extra lines to this file, just add your answers.

For example if your answer to Q0 is: "grep -E Andrew words.txt"

Change the line that starts with

"Q0 answer:"

to

"Q0 answer: grep -E Andrew words.txt"

1) Write a shell pipeline which will print how many classes there are.

Q1 answer: wc -l < classes.tsv

2) Write a shell pipeline which will print how many different courses have classes.

Q2 answer: cut -f1 classes.tsv | sort | uniq | wc -l

3) Write a shell pipeline which will print the course with the most classes, and how many classes are in this course.

Q3 answer: cut -f1 classes.tsv | sort | uniq -c | sort -n | tail -1

4) Write a shell pipeline which will print the two rooms most frequently used by non-LAB CSE classes and how often they are used.

Q4 answer: grep -Fv 'LAB' classes.tsv | cut -f7 | sort | uniq -c | sort -nr | head -2

5) Write a shell pipeline which will print the most common day and hour in the week for classes to start and how many classes start at that time.

Q5 answer: cut -f6 classes.tsv | cut -d'-' -f1 | cut -d':' -f1 | sort | uniq -c | sort -n | tail -1

6) Write a shell pipeline which will print the latest time a class will finish.

Q6 answer: cut -f6 classes.tsv | cut -d' ' -f2 | cut -d'-' -f2 | sort -un | tail -1

7) Write a shell pipeline which will print a list of the course codes of COMP courses that run 2 or more classes of the same type starting at the same time on the same day. (e.g. three tuts starting Monday at 10:00).

Q7 answer: grep -F 'COMP' classes.tsv | cut -f1,3,6 | cut -d'-' -f1 | sort | uniq -d | cut -f1 | sort | uniq

## EXERCISE:

## Editing C Source Files

There is a template file named `editing_programs_answers.txt` which you must use to enter the answers for this exercise.

The autotest scripts depend on the format of `editing_programs_answers.txt` so just add your answers don't otherwise change the file.

The file `program.c` contains a C library implementing some simple sorting algorithms.

1. Write a `sed` command to change all the *functions* from *V1* to *V2*.

### HINT:

It should print:

```

#include "stdlib.h"
#include <stddef.h>
#include "bits/types.h"

typedef int (*compar)(const void *, const void *);

#define SWAP(a, b, size) \
    do { \
        size_t __size = (size); \
        char *__a = (a), *__b = (b); \
        do { \
            char __tmp = *__a; \
            *__a++ = *__b; \
            *__b++ = __tmp; \
        } while (--__size > 0); \
    } while (0)

/**
 * bubble_sort_V2
 * dumb bubble sort using the stdlib::qsort interface
 * @param base    pointer to start of array to be sorted
 * @param nmemb    number of elements in array to be sorted
 * @param size     number of bytes of each element
 * @param comparator function to compare two element
 */
void bubble_sort_V2 (void *base, size_t nmemb, size_t size, compar comparator)
{
    // TODO: use better variable names.
    char *base_ptr = (char *)base;
    for (size_t loop_V1 = 0; loop_V1 < nmemb; loop_V1++) {
        for (size_t loop_V2 = 1; loop_V2 < nmemb; loop_V2++) {
            if ((*comparator)((void *)&base_ptr[loop_V2*size], (void *)&base_ptr[(loop_V2-1)*size])) < 0) {
                SWAP((void *)&base_ptr[loop_V2*size], (void *)&base_ptr[(loop_V2-1)*size],
                    size);
            }
        }
    }
}

extern int strcmp(const char *s1, const char *s2);

int cmpstringp(const void *p1, const void *p2)
{
    return strcmp(*(char *const *)p1, *(char *const *)p2);
}

int cmpintp(const void *p1, const void *p2)
{
    return (*(int *)p1 > *(int *)p2) - (*(int *)p1 < *(int *)p2);
}

extern int printf(const char *format, ...);

int main(void) {
    // Test that our bubble sort is working properly
    int array[10] = {6, 8, 3, 2, 7, 0, 100, -66, 63, 44}; // TODO: make this array bigger

    bubble_sort_V2(array, 10, sizeof(int), cmpintp);

    for (size_t i = 0; i < 10; i++) printf("%d, ", array[i]);
    printf("\n");

    return 0;
}

/**
 * selection_sort_V2
 * selection sort using the stdlib::qsort interface
 * @param base    pointer to start of array to be sorted
 * @param nmemb    number of elements in array to be sorted

```

```

* @param size      number of bytes of each element
* @param comparator function to compare two element
*/
void selection_sort_V2 (void *base, size_t nmemb, size_t size, compar comparator)
{
    // FIXME: implement this function.
    (void) base, (void) nmemb, (void) size, (void) comparator;
    return;
}

/**
* insertion_sort_V2
* insertion sort using the stdlib::qsort interface
* @param base      pointer to start of array to be sorted
* @param nmemb     number of elements in array to be sorted
* @param size      number of bytes of each element
* @param comparator function to compare two element
*/
void insertion_sort_V2 (void *base, size_t nmemb, size_t size, compar comparator)
{
    char *base_ptr = (char *)base;
    for (size_t i = 1; i < nmemb; i++) {
        while ((*comparator)((void *)(&base_ptr[i*size]), (void *)(&base_ptr[(i-1)*size])) < 0)
        {
            SWAP((void *)(&base_ptr[i*size]), (void *)(&base_ptr[(i-1)*size]), size);
        }
    }
}

```

**ANSWER:**

Sample answer:

```
$ sed 's/sort_V1/sort_V2/' program.c
```

Approach:

TODO

As always autotests are available

```
$ 2041 autotest editing_programs Q1
```

2. Write a `sed` command to remove all single line comments starting with `TODO` or `FIXME` .**HINT:**

It should print:

```

#include "stdlib.h"
#include <stddef.h>
#include "bits/types.h"

typedef int (*compar)(const void *, const void *);

#define SWAP(a, b, size) \
    do { \
        size_t __size = (size); \
        char *__a = (a), *__b = (b); \
        do { \
            char __tmp = *__a; \
            *__a++ = *__b; \
            *__b++ = __tmp; \
        } while (--__size > 0); \
    } while (0)

/**
 * bubble_sort_V1
 * dumb bubble sort using the stdlib::qsort interface
 * @param base    pointer to start of array to be sorted
 * @param nmemb    number of elements in array to be sorted
 * @param size     number of bytes of each element
 * @param comparator function to compare two element
 */
void bubble_sort_V1 (void *base, size_t nmemb, size_t size, compar comparator)
{
    char *base_ptr = (char *)base;
    for (size_t loop_V1 = 0; loop_V1 < nmemb; loop_V1++) {
        for (size_t loop_V2 = 1; loop_V2 < nmemb; loop_V2++) {
            if ((*comparator)((void *)&base_ptr[loop_V2*size], (void *)&base_ptr[(loop_V2-1)*size])) < 0) {
                SWAP((void *)&base_ptr[loop_V2*size], (void *)&base_ptr[(loop_V2-1)*size],
                    size);
            }
        }
    }
}

extern int strcmp(const char *s1, const char *s2);

int cmpstringp(const void *p1, const void *p2)
{
    return strcmp(*(char *const *)p1, *(char *const *)p2);
}

int cmpintp(const void *p1, const void *p2)
{
    return (*(int *)p1 > *(int *)p2) - (*(int *)p1 < *(int *)p2);
}

extern int printf(const char *format, ...);

int main(void) {
    // Test that our bubble sort is working properly
    int array[10] = {6, 8, 3, 2, 7, 0, 100, -66, 63, 44};

    bubble_sort_V1(array, 10, sizeof(int), cmpintp);

    for (size_t i = 0; i < 10; i++) printf("%d, ", array[i]);
    printf("\n");

    return 0;
}

/**
 * selection_sort_V1
 * selection sort using the stdlib::qsort interface
 * @param base    pointer to start of array to be sorted
 * @param nmemb    number of elements in array to be sorted

```

```

    * @param size      number of bytes of each element
    * @param comparator function to compare two element
    */
void selection_sort_V1 (void *base, size_t nmemb, size_t size, compar comparator)
{
    (void) base, (void) nmemb, (void) size, (void) comparator;
    return;
}

/**
 * insertion_sort_V1
 * insertion sort using the stdlib::qsort interface
 * @param base      pointer to start of array to be sorted
 * @param nmemb      number of elements in array to be sorted
 * @param size      number of bytes of each element
 * @param comparator function to compare two element
 */
void insertion_sort_V1 (void *base, size_t nmemb, size_t size, compar comparator)
{
    char *base_ptr = (char *)base;
    for (size_t i = 1; i < nmemb; i++) {
        while ((*comparator)((void *)&base_ptr[i*size]), (void *)&base_ptr[(i-1)*size])) < 0)
        {
            SWAP((void *)&base_ptr[i*size], (void *)&base_ptr[(i-1)*size], size);
        }
    }
}

```

**ANSWER:**

Sample answer:

```
$ sed -E 's://\s*(TODO|FIXME).*$::' program.c
```

Approach:

TODO

As always autotests are available

```
$ 2041 autotest editing_programs Q2
```

3. Write a `sed` command to print all lines starting with *extern*.**HINT:**

It should print:

```
extern int strcmp(const char *s1, const char *s2);
extern int printf(const char *format, ...);
```

**ANSWER:**

Sample answer:

```
$ sed -n '/^\s*extern/p' program.c
```

Approach:

TODO

As always autotests are available

```
$ 2041 autotest editing_programs Q3
```

4. write a `sed` command to replace all `include` statements using `with <>`.

HINT:

It should print:



```

#include <stdlib.h>
#include <stddef.h>
#include <bits/types.h>

typedef int (*compar)(const void *, const void *);

#define SWAP(a, b, size) \
    do { \
        size_t __size = (size); \
        char *__a = (a), *__b = (b); \
        do { \
            char __tmp = *__a; \
            *__a++ = *__b; \
            *__b++ = __tmp; \
        } while (--__size > 0); \
    } while (0)

/**
 * bubble_sort_V1
 * dumb bubble sort using the stdlib::qsort interface
 * @param base      pointer to start of array to be sorted
 * @param nmemb      number of elements in array to be sorted
 * @param size       number of bytes of each element
 * @param comparator function to compare two element
 */
void bubble_sort_V1 (void *base, size_t nmemb, size_t size, compar comparator)
{
    // TODO: use better variable names.
    char *base_ptr = (char *)base;
    for (size_t loop_V1 = 0; loop_V1 < nmemb; loop_V1++) {
        for (size_t loop_V2 = 1; loop_V2 < nmemb; loop_V2++) {
            if ((*comparator)((void *)&base_ptr[loop_V2*size], (void *)&base_ptr[(loop_V2-1)*size])) < 0) {
                SWAP((void *)&base_ptr[loop_V2*size], (void *)&base_ptr[(loop_V2-1)*size],
                    size);
            }
        }
    }
}

extern int strcmp(const char *s1, const char *s2);

int cmpstringp(const void *p1, const void *p2)
{
    return strcmp(*(char *const *)p1, *(char *const *)p2);
}

int cmpintp(const void *p1, const void *p2)
{
    return (*(int *)p1 > *(int *)p2) - (*(int *)p1 < *(int *)p2);
}

extern int printf(const char *format, ...);

int main(void) {
    // Test that our bubble sort is working properly
    int array[10] = {6, 8, 3, 2, 7, 0, 100, -66, 63, 44}; // TODO: make this array bigger

    bubble_sort_V1(array, 10, sizeof(int), cmpintp);

    for (size_t i = 0; i < 10; i++) printf("%d, ", array[i]);
    printf("\n");

    return 0;
}

/**
 * selection_sort_V1
 * selection sort using the stdlib::qsort interface
 * @param base      pointer to start of array to be sorted
 * @param nmemb      number of elements in array to be sorted

```

```

* @param size      number of bytes of each element
* @param comparator function to compare two element
*/
void selection_sort_V1 (void *base, size_t nmemb, size_t size, compar comparator)
{
    // FIXME: implement this function.
    (void) base, (void) nmemb, (void) size, (void) comparator;
    return;
}

/**
* insertion_sort_V1
* insertion sort using the stdlib::qsort interface
* @param base      pointer to start of array to be sorted
* @param nmemb      number of elements in array to be sorted
* @param size      number of bytes of each element
* @param comparator function to compare two element
*/
void insertion_sort_V1 (void *base, size_t nmemb, size_t size, compar comparator)
{
    char *base_ptr = (char *)base;
    for (size_t i = 1; i < nmemb; i++) {
        while ((*comparator)((void *)(&base_ptr[i*size]), (void *)(&base_ptr[(i-1)*size])) < 0)
        {
            SWAP((void *)(&base_ptr[i*size]), (void *)(&base_ptr[(i-1)*size]), size);
        }
    }
}

```

**ANSWER:**

Sample answer:

```
$ sed -E 's/^#include\s+"([^\"]*)"#include <\1>/' program.c
```

Approach:

TODO

As always autotests are available

```
$ 2041 autotest editing_programs Q4
```

5. Write a `sed` command to remove the `main` method.**HINT:**

It should print:

```

#include "stdlib.h"
#include <stddef.h>
#include "bits/types.h"

typedef int (*compar)(const void *, const void *);

#define SWAP(a, b, size) \
    do { \
        size_t __size = (size); \
        char *__a = (a), *__b = (b); \
        do { \
            char __tmp = *__a; \
            *__a++ = *__b; \
            *__b++ = __tmp; \
        } while (--__size > 0); \
    } while (0)

/**
 * bubble_sort_V1
 * dumb bubble sort using the stdlib::qsort interface
 * @param base      pointer to start of array to be sorted
 * @param nmemb     number of elements in array to be sorted
 * @param size      number of bytes of each element
 * @param comparator function to compare two element
 */
void bubble_sort_V1 (void *base, size_t nmemb, size_t size, compar comparator)
{
    // TODO: use better variable names.
    char *base_ptr = (char *)base;
    for (size_t loop_V1 = 0; loop_V1 < nmemb; loop_V1++) {
        for (size_t loop_V2 = 1; loop_V2 < nmemb; loop_V2++) {
            if ((*comparator)((void *)&base_ptr[loop_V2*size], (void *)&base_ptr[(loop_V2-1)*size])) < 0) {
                SWAP((void *)&base_ptr[loop_V2*size], (void *)&base_ptr[(loop_V2-1)*size],
                    size);
            }
        }
    }
}

extern int strcmp(const char *s1, const char *s2);

int cmpstringp(const void *p1, const void *p2)
{
    return strcmp(*(char *const *)p1, *(char *const *)p2);
}

int cmpintp(const void *p1, const void *p2)
{
    return (*(int *)p1 > *(int *)p2) - (*(int *)p1 < *(int *)p2);
}

extern int printf(const char *format, ...);

/**
 * selection_sort_V1
 * selection sort using the stdlib::qsort interface
 * @param base      pointer to start of array to be sorted
 * @param nmemb     number of elements in array to be sorted
 * @param size      number of bytes of each element
 * @param comparator function to compare two element
 */
void selection_sort_V1 (void *base, size_t nmemb, size_t size, compar comparator)
{
    // FIXME: implement this function.
    (void) base, (void) nmemb, (void) size, (void) comparator;
    return;
}

/**

```

```

* insertion_sort_V1
* insertion sort using the stdlib::qsort interface
* @param base      pointer to start of array to be sorted
* @param nmemb     number of elements in array to be sorted
* @param size      number of bytes of each element
* @param comparator function to compare two element
*/
void insertion_sort_V1 (void *base, size_t nmemb, size_t size, compar comparator)
{
    char *base_ptr = (char *)base;
    for (size_t i = 1; i < nmemb; i++) {
        while ((*comparator)((void *)&base_ptr[i*size]), (void *)&base_ptr[(i-1)*size])) < 0)
        {
            SWAP((void *)&base_ptr[i*size]), (void *)&base_ptr[(i-1)*size]), size);
        }
    }
}

```

**ANSWER:**

Sample answer:

```
$ sed '/^int main/,/^}/d' program.c
```

Approach:

TODO

As always autotests are available

```
$ 2041 autotest editing_programs Q5
```

When you think your program is working, you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest editing_programs
```

When you are finished working on this exercise, you must submit your work by running `give` :

```
$ give cs2041 lab02_editing_programs editing_programs_answers.txt
```

before **Tuesday 14 June 12:00** to obtain the marks for this lab exercise.

**SOLUTION:**

Sample solution for `editing_programs_answers.txt`

This file is automarked.

Do not add extra lines to this file, just add your answers.

For example if your answer to Q0 is: "grep -E Andrew words.txt"  
Change the line that starts with  
"Q0 answer:"  
to  
"Q0 answer: grep -E Andrew words.txt"

1) Write a sed command to change all the functions from V1 to V2.  
Q1 answer: sed 's/sort\_V1/sort\_V2/' program.c

2) Write a sed command to remove all single line comments starting with TODO or FIXME.  
Q2 answer: sed -E 's://\s\*(TODO|FIXME).\*\$::' program.c

3) Write a sed command to print all lines starting with extern.  
Q3 answer: sed -n '/^\s\*extern/p' program.c

4) Write a sed command to replace all include statements using "" with <>.  
Q4 answer: sed -E 's/^#include\s+"([^\"]\*)"#include <\1>/' program.c

5) Write a sed command to remove the main method.  
Q5 answer: sed '/^int main/,/^}/d' program.c

CHALLENGE EXERCISE:

## Exploring Regular Expression Extensions

There is a template file named `advanced_ab_answers.txt` which you must use to enter the answers for this exercise.

The autotest scripts depend on the format of `advanced_ab_answers.txt` so just add your answers don't otherwise change the file.

Use `grep -P` to test your answers to these questions.

These questions **can't** be solved using the standard regular expression language described in lectures.

The following commands may provide useful information:

```
$ man 1 grep
$ info grep
$ man 7 regex
$ perldoc perlre
```

We've provided a set of test cases in `input.txt`

1. Write a `grep -P` command that prints the lines in a file named `input.txt` containing only the characters `A` and `B` such that there are exactly  $n$  A's followed by exactly  $n$  B's and no other characters.

Matching	Not Matching
AAABBB	AAABB
AB	ABBBBB
AABB	AAAAAA
AAAAAAAAAABBBBBBBBBB	AABBAB

This can't be done with a POSIX regular expression.

You prove this via the the wonderfully named [pumping lemma](#).

It is possible with extensions to regular expressions, e.g. as provided in Perl and PCRE.

Sample answer:

```
$ grep -P '^(A(?1)?B)$' input.txt
```

As always autotests are available

```
$ 2041 autotest advanced_ab
```

When you think your program is working, you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest advanced_ab
```

When you are finished working on this exercise, you must submit your work by running `give` :

```
$ give cs2041 lab02_advanced_ab advanced_ab_answers.txt
```

before **Tuesday 14 June 12:00** to obtain the marks for this lab exercise.

#### SOLUTION:

Sample solution for `advanced_ab_answers.txt`

This file is automarked.

Do not add extra lines to this file, just add your answers.

For example if your answer to Q0 is: "grep -E Andrew words.txt"

Change the line that starts with

"Q0 answer:"

to

"Q0 answer: grep -E Andrew words.txt"

1) Write a `grep -P` command that prints the lines in a file named `input.txt` containing only the characters A and B such that there are exactly n A's followed by exactly n B's and no other characters.

Q1 answer: `grep -P '^(A(?1)?B)$' input.txt`

## Submission

When you are finished each exercises make sure you submit your work by running `give` .

You can run `give` multiple times. Only your last submission will be marked.

Don't submit any exercises you haven't attempted.

If you are working at home, you may find it more convenient to upload your work via [give's web interface](#).

Remember you have until **Week 3 Tuesday 12:00:00** to submit your work.

You cannot obtain marks by e-mailing your code to tutors or lecturers.

You check the files you have submitted [here](#).

Automarking will be run by the lecturer several days after the submission deadline, using test cases different to those `autotest` runs for you. (Hint: do your own testing as well as running `autotest` .)

After automarking is run by the lecturer you can [view your results here](#). The resulting mark will also be available [via give's web interface](#).

## Lab Marks



When all components of a lab are automarked you should be able to view the the marks [via give's web interface](#) or by running this command on a CSE machine:

```
$ 2041 classrun -sturec
```

**COMP(2041|9044) 22T2: Software Construction** is brought to you by  
the [School of Computer Science and Engineering](#)  
at the [University of New South Wales](#), Sydney.

For all enquiries, please email the class account at [cs2041@cse.unsw.edu.au](mailto:cs2041@cse.unsw.edu.au)

CRICOS Provider 00098G