

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 **fake** 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*.

If the student with the lowest *zID* has multiple enrolments, print their enrolment in the course with the highest *Course Code*.

HINT:

It should print:

```
COMP9414|5200002|Tran Joel, Steven Brian Ben  
|3784/3|MECHAH|074.003|23T2|20010111|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 -k1,1r 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 5201804 Wong, Nathan Yuxuan 3778/3 COMPAS 037.997 23T2 20000214 M	
BINF3010 5202234 Zhao, Harry 8543 DPBSA1 073.089 23T2 19970905 M	
BINF3010 5204248 Tan Angus, Andre Oscar Md 3778/1 SENGAH 036.699 23T2 19981217 M	
BINF3010 5206611 Zhang, Ethan Jay Josh 8543 COMPA1 051.641 23T2 19991101 M	
BINF3010 5207861 Chen, Elizabeth Vanessa Michelle 8543 COMPA1 056.589 23T2 19841203 F	
BINF3010 5209846 Zhang, Nicholas 8543 COMPA1 068.062 23T2 20020721 M	
BINF3010 5209994 Gupta, Jesse Jason 3768/4 SENGAH 051.538 23T2 20030205 M	
BINF3010 5212733 Liang, Yifan 3707/4 COMPCS 047.897 23T2 20030116 M	
BINF3010 5217703 Wang, Michael Antonio 8543 COMPA1 051.995 23T2 20021001 M	
BINF3010 5218071 Tan, Angela Kelly Hannah 3784/3 COMPSS 060.166 23T2 19911004 F	
BINF3010 5218274 Chen, Darren Sam Luke 3707/2 DPBSA1 083.761 23T2 19980318 M	
BINF3010 5223005 Phan, Adrian Christopher 3784/3 BIOMDS 067.008 23T2 20031201 M	
BINF3010 5227026 Li, Ryan Haoran Felix 8543 COMPA1 095.598 23T2 20010504 M	
BINF3010 5232551 Qian, Alexander Ibrahim Louis 3707/1 ELECAH 072.755 23T2 19921123 M	
BINF3010 5237535 Nguyen, Michael 3778/3 COMPCS 057.865 23T2 20030624 M	
BINF3010 5237727 Cao, Cheng Jordan Zac ELECAH 068.501 23T2 19980724 M	3762/2 BIOMDS
BINF3010 5239503 Liu, Jessica 8543 COMPA1 037.452 23T2 19941212 F	
BINF3010 5240138 Lim, Tristan Ivan Tsz Jacob 3785/2 COMPY1 039.284 23T2 19970720 M	
BINF3010 5244983 Li, Ryan Joel 8543 COMPA1 080.377 23T2 19970304 M	
BINF3010 5245695 He, Chun 7004/1 ENGGAH 067.036 23T2 20010207 M	
BINF3010 5245941 Xu, Haotian Arnav Leo 3768/1 SENGAH 041.633 23T2 20050405 M	
BINF3010 5246049 He, Peter 7004/1 SENGAH 087.590 23T2 19980816 M	
BINF3010 5248646 Wang, Jessica Rachel 8543 COMPA1 045.217 23T2 20031118 F	
BINF3010 5249320 Du, Amy Jing Yu MECHAH 073.698 23T2 19990312 F	3133/4 BIOMDS
BINF3010 5249555 Wang, Eric Yang 8543 COMPA1 067.363 23T2 19960220 M	
BINF3010 5251112 Huang, Luke Isaac Ruiqi 3778/1 SENGAH 082.871 23T2 19981025 M	
BINF3010 5251256 Smith, Liam Bohan 3785/1 ELECAH 060.653 23T2 19970916 M	
BINF3010 5252013 Zhao, Isaac 3784/1 COMPAS 066.212 23T2 19980930 M	
BINF3010 5253443 Song Yuhan, Alice Winnie 3785/2 DPENX1 032.704 23T2 20030911 F	
BINF3010 5257134 Chan, Hannah 3778/2 SENGAH 073.933 23T2 20030202 F	
BINF3010 5258182 Fan, Tim Eric COMPSS 073.095 23T2 20010925 M	7004/1 COMPAS
BINF3010 5260129 Huynh, Karen 3785/3 ELECAH 073.287 23T2 20000512 F	
BINF3010 5263438 Kim Michael, Liam Timothy 3764/1 COMPCH 046.753 23T2 20030213 M	
BINF3010 5263620 Yang, Ryan Ze 3707/1 COMPA1 070.901 23T2 19981021 M	
BINF3010 5264102 Chen, Alex Chris Gabriel 8543 COMPA1 080.160 23T2 19990605 M	
BINF3010 5266002 Ma, Adrian	

1650 COMPY1 019.855 23T2 20010522 M	
BINF3010 5267625 Chan, Mitchell Peter Harrison Gordon	
8543 MTRNAH 065.261 23T2 19960526 M	
BINF3010 5268496 Chen, Yi Sebastian	
3789/4 COMPAS 052.445 23T2 19920629 M	
BINF3010 5272109 Lin Aditi, Catherine Vivian	
3707/4 COMPSS 096.447 23T2 20040903 F	
BINF3010 5272404 Nguyen, Dylan Benjamin Tristan	
1650 COMPA1 032.872 23T2 20000416 M	
BINF3010 5276965 Park, Tony Anish	3674/2 UNDL-
U 050.429 23T2 20010310 M	
BINF3010 5289021 Wang, Ryan Jonathan Gordon	
8543 COMPA1 073.166 23T2 19990202 M	
BINF3010 5290699 Hu, Luke Ivan Danny	
3785/1 COMPA1 046.852 23T2 19991029 M	
BINF3010 5295319 Wang, Michael Jia	
8543 COMPA1 058.703 23T2 19921120 M	
BINF3010 5297819 Nguyen, Ryan Josh Ayush	
3707/1 COMPA1 052.193 23T2 20001111 M	
BINF9010 5202701 Yin, Xin Jessica	
3783/2 CEICAH 061.013 23T2 19861112 F	
BINF9010 5228372 Wang, William Felix	
8543 COMPA1 051.552 23T2 20000307 M	
BINF9010 5229328 Xu, Alexander Ricky Nathaniel Zac Matthew	
3778/3 COMPCS 043.644 23T2 20010904 M	
BINF9010 5232791 Ding, Derek Chun Will	3783/1 ACTLE1
COMPA1 080.907 23T2 19991228 M	
BINF9010 5266930 Le, Andrew Jimmy Kenneth Mohammed	
3778/3 COMPAS 083.830 23T2 19950921 M	
BIOM1010 5205212 Zhang Yuxuan, Jason Sean Bowen Zhou	
8543 COMPA1 089.914 23T2 20040220 M	
BIOM1010 5206606 Wang, Jessica Katherine Emily Sarah	
3707/1 COMPA1 049.001 23T2 20000309 F	
BIOM1010 5206630 Xiong Sophie, Jessica Winnie Olivia	3738/1 COMPD1
FINSA1 037.897 23T2 19901113 F	
BIOM1010 5206723 Patel, Jiahao Angus	
3674/1 COMPQS 028.951 23T2 19960526 M	
BIOM1010 5207058 Lu, James Nathaniel Jun	
3784/2 COMPCS 036.779 23T2 20020209 M	
BIOM1010 5207762 Yu, Xin Sophie Ashley	
1650 MECHAH 057.132 23T2 19990127 F	
BIOM1010 5207793 Huang, Aiden Laurence	
8543 DPHUB1 065.937 23T2 20020526 M	
BIOM1010 5210902 Hua, Jimmy Danny	
3736/1 CRIMA1 069.992 23T2 20030329 M	
BIOM1010 5213983 Yin, Xinyi Jiayu Amanda	
1650 AEROAH 069.873 23T2 20040806 F	
BIOM1010 5217184 Luo, Tiffany Grace	3970/1 COMPA1
MTRNAH 078.933 23T2 19991211 F	
BIOM1010 5219063 Huynh, Jordan Cheng	3673/4 ACTLD1
COMPA1 067.054 23T2 20040201 M	
BIOM1010 5221130 Xu Xin, Megan Yan	
3778/2 COMPA1 047.261 23T2 19900504 F	
BIOM1010 5222251 Fu, Zhou	3767/3 COMPS1
TELEAH 079.677 23T2 19971002 M	
BIOM1010 5222829 Khan, Tim Stephen	
8959 MTRNES 039.141 23T2 20011022 M	
BIOM1010 5223425 Feng, Lachlan Jia	
3706/1 BINFAH 029.145 23T2 20000828 M	
BIOM1010 5225537 Wu, Han Christopher	
3707/1 COMPA1 046.467 23T2 19891023 M	
BIOM1010 5226090 Fu Jenny, Natalie Tiffany Ashley	
3784/1 COMPZ1 059.421 23T2 20000725 F	
BIOM1010 5227987 Luo, Brandon Jack	3707/2 COMPAS
COMPCS 074.718 23T2 20031109 M	
BIOM1010 5234944 Xu, Tom	
3785/1 DPBSA1 071.419 23T2 19990921 M	
BIOM1010 5235284 Luo, Zihao Jian Isaac	
3707/4 COMPCS 047.997 23T2 20050721 M	
BIOM1010 5240207 Tang, Winnie Xinyue	3768/4 UNDL-
U 085.521 23T2 19991214 F	
BIOM1010 5244252 Liu Katherine, Jessica	

8543 COMPA1 088.472 23T2 20050812 F	
BIOM1010 5244507 Kim, Han Ryan	
3673/2 COMPES 066.370 23T2 19960115 M	
BIOM1010 5244938 Zhang, Jessica Vanessa	
8543 COMPA1 093.235 23T2 20000315 F	
BIOM1010 5245298 Lu, Andrew Jiawei	
3784/1 MTRNAH 042.124 23T2 19990809 M	
BIOM1010 5246919 Yu, Andy Gabriel	
3784/1 MECHAH 057.784 23T2 20040801 M	
BIOM1010 5247276 Ma, Dylan	
3778/1 COMPA1 028.814 23T2 20020906 M	
BIOM1010 5248871 Zou William, Jiayi Andrew Hugo Yang	
3783/1 COMPDI 066.684 23T2 20010720 M	
BIOM1010 5249915 Liu, Andy Jordan	
8543 COMPA1 052.561 23T2 19980927 M	
BIOM1010 5250595 Yang Vivian, Angela	
3778/2 COMPA1 058.900 23T2 20001105 F	
BIOM1010 5251436 Shen, Mohammad Laurence	
7003/1 SENGAH 068.050 23T2 20000710 M	
BIOM1010 5252076 Ma, Benjamin George	
7001/1 COMPDI 019.147 23T2 20001203 M	
BIOM1010 5252677 Wang, Andrew Max	
8543 COMPA1 031.697 23T2 19960927 M	
BIOM1010 5255168 Nguyen Robert, Luke Henry	3961/1 COMPAS
INFSKS 062.165 23T2 19871029 M	
BIOM1010 5255953 Yu, Hannah	3778/2 UNDL-
U 037.419 23T2 20040128 F	
BIOM1010 5255975 Duan, Ray Jian Jonathan	3962/1 COMMJ1
COMPDI 046.259 23T2 19940727 M	
BIOM1010 5258536 Ng, Zihao Austin Ayush	
3674/2 ELECAH 055.967 23T2 19950803 M	
BIOM1010 5259131 Zhang Charlotte Elizabeth, Anna Alicia Lara	
Christ 8543 COMPA1 066.881 23T2 20020612 F	
BIOM1010 5264771 Lim, David	
3674/1 DPENX1 066.111 23T2 19991026 M	
BIOM1010 5267838 Fu, Cindy Hannah	3061/1 DDESB1
MDIAR2 030.681 23T2 20020326 F	
BIOM1010 5269780 Zhang, Angela Yan	
8543 COMPA1 084.410 23T2 20010429 F	
BIOM1010 5270784 Liao Jerry, Jonathan Liam	8338 COMPA1
MATHA1 082.284 23T2 20040625 M	
BIOM1010 5275340 Wang Thomas, Matthew	
3707/1 COMPA1 065.881 23T2 19870726 M	
BIOM1010 5275746 Gong, Timothy Hugo	
3768/3 CEICAH 032.200 23T2 19860410 M	
BIOM1010 5279455 Yang, Jack Joel	
3784/2 COMPER 090.653 23T2 20000407 M	
BIOM1010 5279645 Wang, William	
8543 COMPA1 079.799 23T2 19800122 M	
BIOM1010 5280127 Tang Andy, Brian Dylan	
3707/4 COMPSS 043.828 23T2 20040213 M	
BIOM1010 5282285 Tao, Steven	
8750/1 MATHNS 057.091 23T2 19910919 M	
BIOM1010 5282675 Lai, Yu Emma Lara	
3782/1 BINFAH 099.858 23T2 20040612 F	
BIOM1010 5283612 Khan Jessica, Lucy Xinyue Michelle	
3784/3 DPENX1 080.405 23T2 20040921 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 xth column and stop sorting end of yth 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:

COMP9313|5282134|Li, Christopher Oliver Joel
|3778/2|COMPCS|061.069|23T2|19570918|M
COMP9417|5289452|Nguyen, Joshua Terry
|3778/2|COMPCS|041.749|23T2|19570918|M
DPGE1002|5289452|Nguyen, Joshua Terry
|3778/2|COMPCS|041.749|23T2|19570918|M
COMP1511|5236538|Trinh, Yan Natalie
|3707/1|COMPBH|021.732|23T2|19590220|F
COMP4952|5236538|Trinh, Yan Natalie
|3707/1|COMPBH|021.732|23T2|19590220|F
COMP9417|5284939|Mei, Frank Austin
|3764/2|COMPA1
MTRNAH|067.113|23T2|19590220|M
COMP9900|5250145|Zhao, Victor Gordon
|3778/1|COMPCS|083.041|23T2|19590220|M
DPBS1120|5284939|Mei, Frank Austin
|3764/2|COMPA1
MTRNAH|067.113|23T2|19590220|M
COMP3153|5276435|Liu, Grace Joyce
|8543|COMPA1|061.354|23T2|19620722|F
COMP6452|5276435|Liu, Grace Joyce
|8543|COMPA1|061.354|23T2|19620722|F
COMP9312|5276435|Liu, Grace Joyce
|8543|COMPA1|061.354|23T2|19620722|F
COMP9444|5286618|Wang, Victoria Aditi Jing Jennifer Alicia
Jennifer|3707/1|ELECAH|045.011|23T2|19620722|F
DPGE1002|5240982|Xiao Tiffany, Emily
|3673/3|COMPI1
ELECAH|043.846|23T2|19620722|F
COMP1531|5287908|Mu, Dhruv James Patrick
|3782/4|ACCTA1
FINSA1|061.189|23T2|19630823|M
COMP4601|5215280|Huang, Richard Hugo Isaac
|3778/3|COMPA1|087.705|23T2|19630823|M
COMP9024|5287908|Mu, Dhruv James Patrick
|3782/4|ACCTA1
FINSA1|061.189|23T2|19630823|M
COMP9417|5215280|Huang, Richard Hugo Isaac
|3778/3|COMPA1|087.705|23T2|19630823|M
COMP9444|5263308|Lee, Chris Louis
|8543|COMPA1|056.191|23T2|19630823|M
DPGE1001|5243700|Wei, Adrian
|3791/3|ELECAH|093.170|23T2|19630823|M
DPGE1002|5215280|Huang, Richard Hugo Isaac
|3778/3|COMPA1|087.705|23T2|19630823|M
DPST1031|5209031|Nguyen Lisa, Sarah Christine Amy Jiayu Emma
|8543|COMPA1|074.454|23T2|19630823|F
COMP1531|5217454|Lee, Xin Victoria
|3778/2|COMPA1|084.053|23T2|19661218|F
COMP9021|5217454|Lee, Xin Victoria
|3778/2|COMPA1|084.053|23T2|19661218|F
COMP9024|5263205|Li Gregory, Oliver
|3785/1|COMPCS|070.453|23T2|19661218|M
COMP9312|5263205|Li Gregory, Oliver
|3785/1|COMPCS|070.453|23T2|19661218|M
COMP1511|5278327|Kim, Hao Ray
|3778/1|SENGAH|074.695|23T2|19681109|M
COMP1521|5249383|Li, Luke
|3778/2|COMPA1|052.228|23T2|19681109|M
COMP1911|5222418|Yang, Matthew Nicholas
|8543|COMPCS|070.222|23T2|19681109|M
COMP2041|5278327|Kim, Hao Ray
|3778/1|SENGAH|074.695|23T2|19681109|M
COMP2521|5249383|Li, Luke
|3778/2|COMPA1|052.228|23T2|19681109|M
COMP3121|5280978|Zhang, Emily Charlotte Catherine Chloe Christine
|8543|COMPA1|056.223|23T2|19681109|F
COMP3900|5280978|Zhang, Emily Charlotte Catherine Chloe Christine
|8543|COMPA1|056.223|23T2|19681109|F
COMP6452|5249383|Li, Luke
|3778/2|COMPA1|052.228|23T2|19681109|M
COMP9331|5280978|Zhang, Emily Charlotte Catherine Chloe Christine
|8543|COMPA1|056.223|23T2|19681109|F
COMP9417|5278327|Kim, Hao Ray
|3778/1|SENGAH|074.695|23T2|19681109|M
DPBS1120|5280433|Chen Rui, Aryan

3778/3 COMPAS 082.709 23T2 19681109 M	
DPST1013 5280978 Zhang, Emily Charlotte Catherine Chloe Christine	
8543 COMPA1 056.223 23T2 19681109 F	
COMP1511 5265057 Wan, Chun Jin Samuel	
3767/3 DPHUD1 038.415 23T2 19690307 M	
COMP1521 5201661 Yang, Joshua Tsz David	
3778/1 SENGAH 066.655 23T2 19690307 M	
COMP1521 5289463 Zhao, Andrew Karan Arjun	
7004/1 SENGAH 064.292 23T2 19690307 M	
COMP1531 5216127 Geng, Yan Ken	
3956/1 BIOMBS 080.077 23T2 19690307 M	
COMP2041 5201661 Yang, Joshua Tsz David	
3778/1 SENGAH 066.655 23T2 19690307 M	
COMP3141 5265057 Wan, Chun Jin Samuel	
3767/3 DPHUD1 038.415 23T2 19690307 M	
COMP3141 5289463 Zhao, Andrew Karan Arjun	
7004/1 SENGAH 064.292 23T2 19690307 M	
COMP9020 5289463 Zhao, Andrew Karan Arjun	
7004/1 SENGAH 064.292 23T2 19690307 M	
COMP9414 5216127 Geng, Yan Ken	
3956/1 BIOMBS 080.077 23T2 19690307 M	
COMP1521 5219286 Guo, Joyce Karen	
8338 COMPAS 067.357 23T2 19691224 F	
COMP1911 5250938 Lin Zheng, Oliver	3959/3 COMPAS
COMPSS 060.545 23T2 19691224 M	
COMP3511 5250938 Lin Zheng, Oliver	3959/3 COMPAS
COMPSS 060.545 23T2 19691224 M	
COMP1511 5268502 Hoang, Stephen Tao Minh Samuel	1650 ECONI1
MTRNAH 057.305 23T2 19700119 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:

COMP1511|5212432|Wang, Chloe Amanda Yu
 |8543|COMPA1|062.999|23T2|19960408|F
 COMP9313|5215738|Nguyen, Max
 |3707/1|COMPA1|081.999|23T2|19920915|M
 COMP9417|5215738|Nguyen, Max
 |3707/1|COMPA1|081.999|23T2|19920915|M
 COMP3141|5219983|Zhu, Rachel Grace
 |8543|DPBSA1|078.999|23T2|19980225|F
 DPGE1002|5219983|Zhu, Rachel Grace
 |8543|DPBSA1|078.999|23T2|19980225|F
 DPBS1150|5225868|Wong, Yu Xinyue
 |3707/1|SENGAH|062.999|23T2|19940626|F
 COMP9444|5226001|Sharma, Aryan Zachary Francis
 |3768/3|ELECAH|078.999|23T2|19871129|M
 DPGE1003|5226001|Sharma, Aryan Zachary Francis
 |3768/3|ELECAH|078.999|23T2|19871129|M
 COMP2041|5235855|Jiang, Jesse Pranav
 |3768/4|COMPAH|058.999|23T2|19990220|M
 COMP1511|5250497|Liu, Emily Karen
 |8543|COMPA1|060.999|23T2|20010423|F
 COMP1531|5250497|Liu, Emily Karen
 |8543|COMPA1|060.999|23T2|20010423|F
 COMP9444|5250497|Liu, Emily Karen
 |8543|COMPA1|060.999|23T2|20010423|F
 DPBS1140|5250497|Liu, Emily Karen
 |8543|COMPA1|060.999|23T2|20010423|F
 COMP9444|5251974|Lee, Nicole Rachel
 |7004/1|COMPAS|048.999|23T2|19970104|F
 COMP3121|5263804|Huang, Alice
 |3778/3|COMPAS|034.999|23T2|19971203|F
 DPGE1003|5263804|Huang, Alice
 |3778/3|COMPAS|034.999|23T2|19971203|F
 COMP1521|5281281|Wang, Anna Sophia
 |8543|COMPA1|010.999|23T2|20050530|F
 COMP3900|5281281|Wang, Anna Sophia
 |8543|COMPA1|010.999|23T2|20050530|F
 DPGE1002|5281281|Wang, Anna Sophia
 |8543|COMPA1|010.999|23T2|20050530|F
 COMP1531|5215378|Wu, Samuel Zihao
 |3778/2|COMPCS|096.998|23T2|19940626|M
 COMP6452|5251820|Guo, Zheng
 |3781/1|AEROAH|084.998|23T2|20031228|M
 COMP2521|5252367|Zhao, Edward
 |3707/2|MTRNAH|035.998|23T2|20050629|M
 COMP1511|5258610|Li, Daniel Yuhao
 |8543|COMPA1|071.998|23T2|19901016|M
 COMP1521|5262296|Qi, Zhou Yan Callum Yang
 |3784/2|MATHP1|064.998|23T2|20000629|M
 COMP2511|5262296|Qi, Zhou Yan Callum Yang
 |3784/2|MATHP1|064.998|23T2|20000629|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 13 June 12:00 (midday)** (2023-06-13 12:00:00) 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 -k1,1r 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

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:

1275

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:

75

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:

```
84 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:

```
92 Online
28 Quad G054
```

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 in the week and hour in the day 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:

```
48 Tue 14
```

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:

```
21
```

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:

```
COMP1511
COMP1521
COMP1531
COMP1911
COMP2041
COMP2511
COMP2521
COMP3121
COMP3331
COMP3511
COMP3900
COMP6445
COMP6452
COMP6733
COMP6845
COMP9044
COMP9101
COMP9311
COMP9312
COMP9313
COMP9331
COMP9414
COMP9417
COMP9444
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 13 June 12:00 (midday)** (2023-06-13 12:00:00) 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 `sort` related *functions* from *V1* to *V2*.

This includes all relevant comments.

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 13 June 12:00 (midday)** (2023-06-13 12:00:00) 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/^\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

ANSWER:

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 13 June 12:00 (midday)** (2023-06-13 12:00:00) 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 (midday)** 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) 23T2: 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
CRICOS Provider 00098G