

Assignment 2

Due Date

The assignment is due at 3PM Wednesday May 1st 2019



Background

You should be paying attention in class, but, oh, the Internet is *such* a distraction. What with Governments to hack, a Census to fail, Google to search, cat pictures to peruse, YouTube to waste time on, and Snapchat to send coded messages on, there's very little University wifi bandwidth left for actual work. Why is that?

I've identified a way to examine which web sites students visit when they're using their laptops on campus. We'll look at just a 24-hour period of these and discover what people are up to based on sessions they initiate with particular web servers. You'll have the date and time, the web server name, and the IP address of the student's laptop.

You will need to store data in collections organised by server. The collection should be ordered in a first-come-first-served manner, i.e. the first server identified would be the first group in the collection, the second server identified would be second, and so on. Each group would store the name of the server and a cluster of session information. Each cluster of sessions should be ordered chronologically, i.e. earliest times first, latest times last.

You will need to provide some summary data (e.g. number of sessions per hour and total number of sessions per server), and an ability to display all sessions containing particular text (e.g. source IP address, date/time, or destination).

Task

Based *only* on the information above:

- a Which underlying data structure (*array* or *linked-list*) will you use as a basis to model the overall collection of sessions? In two–three sentences, justify your answer.
- b Which kind of abstract data type (*binary tree*, *general tree*, *array*, *stack*, *priority queue*, *double-ended queue*, *set*, *list*, etc.) would you use to model the collection? In two–three sentences, justify your answer by indicating the functionality required.
- c Which underlying data structure (*array* or *linked-list*) will you use as a basis to model the cluster of sessions for each server? In two–three sentences, justify your answer.
- d Which kind of abstract data type (*binary tree*, *general tree*, *array*, *stack*, *priority queue*, *double-ended queue*, *set*, *list*, etc.) would you use to model the cluster of sessions for each server? In two–three sentences, justify your answer by indicating the functionality required.

The types required will now be presented.

A session is defined as follows:

```
struct session_int {
    char date[20];
    char source_IP[16];
    char destination[50];
};
typedef struct session_int *session;
```

and you may assume the existence of those types and the following functions:

```
void init_session(session *sp, char *when, char *from,
    char *to);
char *get_date(session s);
char *get_source_IP(session s);
char *get_destination (session s);
void set_date(session s, char *when);
void set_source_IP(session s, char *from);
void set_destination(session s, char *to);
char *to_string(session s);
```

A server is defined as follows:

```
struct server_int {
    char name[50];
    cluster sessions;
};
```

```
typedef server_int *server;
```

and you may assume the existence of those types and the following functions:

```
void init_server(server *sp, char *who);
char *get_name(server s);
cluster get_sessions(server s);
void set_name(server s, char *who);
void set_sessions(server s, cluster c);
```

A cluster is a group of sessions. You must define the cluster type based upon your answer to (c) above. For example:

```
typedef ??? cluster;
```

You must also define what the type collection is based upon your answers to (a). This will be a group of servers (as defined above) which consist of the server's name and a cluster. For example:

```
typedef ??? collection;
```

Finally, with those types defined, a variable (called archive in the below) can be defined to be of type collection.

```
collection archive;
```

- e You must complete `assig_twol19.h` and `assig_twol19.c` by adding the *cluster*, and *collection* types from above, and also the functions to store sessions in your cluster, servers in your collection, and to search and display the results. Function stubs are already present; you must complete these and may add others.

If your answer to (a) or (c) above is a linked-list then *you will also need to write* `node.h` and `node.c` and add them to the project.

Provisions

A Visual Studio project is available on MyLO for you to download and use as a starting point. This comprises the following files:

- `session.h` and `session.c` — the *Session* ADT as specified above. *These files are complete;*
- `server.h` and `server.c` — the *Server* ADT as specified above. *These files are complete;*
- `assig_twol19.h` — the file into which you should declare the `cluster` and `collection` types; and
- `assig_twol19.c` — the file which contains the `main()` function and other functions which you write to implement the required task.

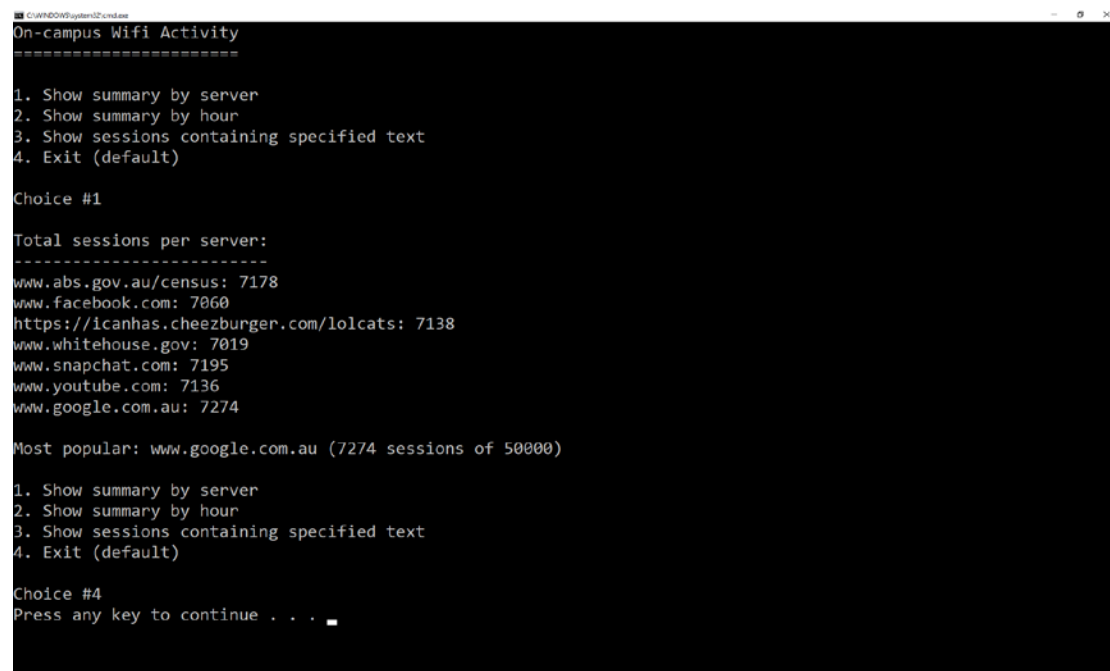
Program specification

First you must obtain the data. (In this case the data is generated by a function which is provided.) This data must be stored (one session at a time) in appropriate collections. At the top level there should be a collection of servers and for each of these servers there should be a cluster of the sessions (stored in ascending order by date and time).

Once the data have been stored in the collections, a menu should be presented offering four choices. The user should be able to view summary information by server, summary information by hour, all sessions containing specified text, or to quit. The code to manage this menu is provided.

Hint: the provided code contains a function, `to_lower()`, to convert text to lower-case so that case-insensitive comparison may be made. The comparison can be made using `strstr()` which looks for its second parameter within its first. You'll need to check the date, source IP address, and destination (server name). You may also find `strtok()` and `strcpy()` from `<string.h>` and `atoi()` from `<stdlib.h>` useful functions.

Sample output (see below) illustrates the required behaviour:



```
On-campus Wifi Activity
=====
1. Show summary by server
2. Show summary by hour
3. Show sessions containing specified text
4. Exit (default)

Choice #1

Total sessions per server:
-----
www.abs.gov.au/census: 7178
www.facebook.com: 7060
https://icanhas.cheezburger.com/lolcats: 7138
www.whitehouse.gov: 7019
www.snapchat.com: 7195
www.youtube.com: 7136
www.google.com.au: 7274

Most popular: www.google.com.au (7274 sessions of 50000)

1. Show summary by server
2. Show summary by hour
3. Show sessions containing specified text
4. Exit (default)

Choice #4
Press any key to continue . . .
```

```
C:\WINDOWS\system32\cmd.exe
=====
1. Show summary by server
2. Show summary by hour
3. Show sessions containing specified text
4. Exit (default)

Choice #2
Total sessions per hour:
-----
00: 2075          01: 2065
02: 2019          03: 2031
04: 2050          05: 2169
06: 2121          07: 2067
08: 2051          09: 2077
10: 2124          11: 2032
12: 2051          13: 2145
14: 2103          15: 2174
16: 2047          17: 2112
18: 2135          19: 2068
20: 2105          21: 2079
22: 2050          23: 2050

Most popular: 15 hours (2174 sessions of 50000)

1. Show summary by server
2. Show summary by hour
3. Show sessions containing specified text
4. Exit (default)

Choice #
_

C:\WINDOWS\system32\cmd.exe
1. Show summary by server
2. Show summary by hour
3. Show sessions containing specified text
4. Exit (default)

Choice #3

Enter text: 00:59:00

Showing all sessions containing "00:59:00":
-----
At 2019/03/28 00:59:00, session initiated from: 010.100.241.090 to www.snapchat.com
At 2019/03/28 00:59:00, session initiated from: 010.100.229.255 to www.youtube.com

1. Show summary by server
2. Show summary by hour
3. Show sessions containing specified text
4. Exit (default)

Choice #3

Enter text: snapchat

Showing all sessions containing "snapchat":
-----
At 2019/03/28 00:00:10, session initiated from: 010.100.254.026 to www.snapchat.com
At 2019/03/28 00:00:26, session initiated from: 010.100.247.238 to www.snapchat.com
At 2019/03/28 00:00:26, session initiated from: 010.100.239.115 to www.snapchat.com
At 2019/03/28 00:00:28, session initiated from: 010.100.228.212 to www.snapchat.com
At 2019/03/28 00:00:31, session initiated from: 010.100.232.024 to www.snapchat.com
```

```
C:\WINDOWS\system32\cmd.exe
At 2019/03/28 23:59:07, session initiated from: 010.100.254.103 to www.snapchat.com
At 2019/03/28 23:59:11, session initiated from: 010.100.236.196 to www.snapchat.com
At 2019/03/28 23:59:14, session initiated from: 010.100.226.064 to www.snapchat.com
At 2019/03/28 23:59:25, session initiated from: 010.100.250.050 to www.snapchat.com

1. Show summary by server
2. Show summary by hour
3. Show sessions containing specified text
4. Exit (default)

Choice #3

Enter text: 10.100.228.155

Showing all sessions containing "10.100.228.155":
-----
At 2019/03/28 13:31:32, session initiated from: 010.100.228.155 to www.abs.gov.au/census
At 2019/03/28 15:17:07, session initiated from: 010.100.228.155 to www.abs.gov.au/census
At 2019/03/28 10:10:18, session initiated from: 010.100.228.155 to www.whitehouse.gov
At 2019/03/28 23:04:28, session initiated from: 010.100.228.155 to www.whitehouse.gov
At 2019/03/28 05:26:50, session initiated from: 010.100.228.155 to www.snapchat.com
At 2019/03/28 19:43:47, session initiated from: 010.100.228.155 to www.youtube.com
At 2019/03/28 09:51:56, session initiated from: 010.100.228.155 to www.google.com.au
At 2019/03/28 14:48:57, session initiated from: 010.100.228.155 to www.google.com.au

1. Show summary by server
2. Show summary by hour
3. Show sessions containing specified text
4. Exit (default)

Choice #_
```

Program Style

The program you write for this assignment must be contained in six–eight files as follows:

- the six files provided; and
- `node.h` and `node.c` for nodes of linked lists (if this data structure has been selected).

Your program should follow the following coding conventions:

- no function should be longer than about half a screen-full or so of code;
- `const` variable identifiers should be used as much as possible, should be written all in upper case and should be declared before all other variables;
- `#defined` symbols should be used for constant values only if `const` is inappropriate
- global variables should be used sparingly with parameter lists used to pass information in and out of functions.
- local variables should only be defined at the beginning of functions and their identifiers should start with a lower case letter;
- every `if` and `if-else` statement should have a block of code (i.e. collections of lines surrounded by `{` and `}`) for both the `if` part and the `else` part (if used)
- every `do`, `for`, and `while` loop should have a block of code (i.e. `{ }`s)
- the keyword `continue` should not be used;
- the keyword `break` should only be used as part of a `switch` statement;
- opening and closing braces of a block should be aligned;
- all code within a block should be aligned and indented 1 tab stop (or 4 spaces) from the braces marking this block;
- commenting:
 - There should be a block of header comment which includes at least
 - file name
 - student name

- student identity number
- a statement of the purpose of the program
- date
- Each variable declaration should be commented
- There should be a comment identifying groups of statements that do various parts of the task
- Comments should describe the strategy of the code and should not simply translate the C into English

What and how to submit

What to submit

You must make *both* a paper and an electronic submission.

Paper submission

- A paper cover page (blanks can be collected from the Information and Communications Technology Discipline Help Desk).
- Written answers to parts (a)–(d) above.
- A *landscape oriented* print-out of `assig_two119.h` and `assig_two119.c` and any other program files you have added to the solution. *Your assignment will not be fully marked unless these are present.*

Electronic submission

- You should submit the entire Visual Studio project folder compressed as a ZIP file.

How to submit

Paper submission

- Firmly staple together all of the required documents (with the signed cover page on top) and place them in the appropriate submissions box near the ICT Help Desk.

Electronic submission

- You should ZIP the Visual Studio project folder and then submit it to the “Assignment 2” assignment drop-box on *KIT107*’s MyLO site.

Marking scheme

Task/Topic	Maximum mark
<i>Design</i>	
ADTs chosen wisely	3
Data structures correct and justified	3
<i>Program operates as specified</i>	
node.c correctly completed (if required, or array equivalent)	6
assig_twol19.c correctly completed —	
• initialise_archive() to initialise the archive variable	2
• add_existing() to	
o add a session to a non-empty cluster	2
o in chronological order	2
• add_session() to	
o add a session to an empty archive	1
o add a session to a new server (adding the server in first-in-first-out order)	2
o call add_existing() to add a session to a non-empty cluster	1
• show_servers() to display the summary for menu option 1	4
• show_times() to display the summary for menu option 2	4
• show_selection() to display all relevant sessions for menu option 3	6
<i>Program Style</i>	
Does not unnecessarily repeat tests or have other redundant/confusing code	4
Uses correctly the C naming conventions	4
Alignment of code and use of white space makes code readable	4
Always uses blocks in branch and loop constructs	4
Meaningful identifiers	4
Variables defined at the start of functions	4
Header comments for the program (author, date etc.)	4
Each variable declaration is commented	4
Comments within the code indicate the purpose of sections of code (but DO NOT just duplicate what the code says)	4

Plagiarism and Cheating:

Practical assignments are used in the ICT discipline for students to both reinforce and demonstrate their understanding of material which has been presented in class. They have a role both for assessment and for learning. It is a requirement that work you hand in for assessment is substantially your own.

Working with others

One effective way to grasp principles and concepts is to discuss the issues with your peers and/or friends. You are encouraged to do this. We also encourage you to discuss aspects of practical assignments with others. However, once you have clarified the principles, you must express them in writing or electronically entirely by yourself. In other words you must develop the algorithm to solve the problem and write the program which implements this algorithm alone and with the help of no one else (other than staff). You can discuss the problem with other students, but not how to solve it.

Cheating

- Cheating occurs if you claim work as your own when it is substantially the work of someone else.
- Cheating is an offence under the [Ordinance of Student Discipline](#) within the University. Furthermore, the ICT profession has ethical standards in which cheating has no place.
- Cheating involves two or more parties.
 - If you allow written work, computer listings, or electronic version of your code to be borrowed or copied by another student you are an equal partner in the act of cheating.
 - You should be careful to ensure that your work is not left in a situation where it may be stolen by others.
- Where there is a reasonable cause to believe that a case of cheating has occurred, this will be brought to the attention of the unit lecturer. If the lecturer considers that there is evidence of cheating, then no marks will be given to any of the students involved. The case will be referred to the Head of School for consideration of further action.

Julian Dermoudy, March 30th 2019.