

- Course: CSC340.01
- Student: <Keith> <Rich>
- Teammate: <George> <Last Name>
- Assignment Number: 01
- Assignment Due Date & Time: 07-02-2018 at 11:59 PM

## - PART 1: Tic Tac Toe

Provided was the header and main. We had to implement makeMove, isWin, isDraw, and displayBoard.

The image displays two screenshots of a C++ IDE (Visual Studio Code) showing the implementation of a Tic Tac Toe game. The code is written in C++ and includes a main function and a makeMove function. The game board is represented as a 3x3 array of characters (X, O, or empty space). The game logic includes checking for a win, a draw, or a valid move.

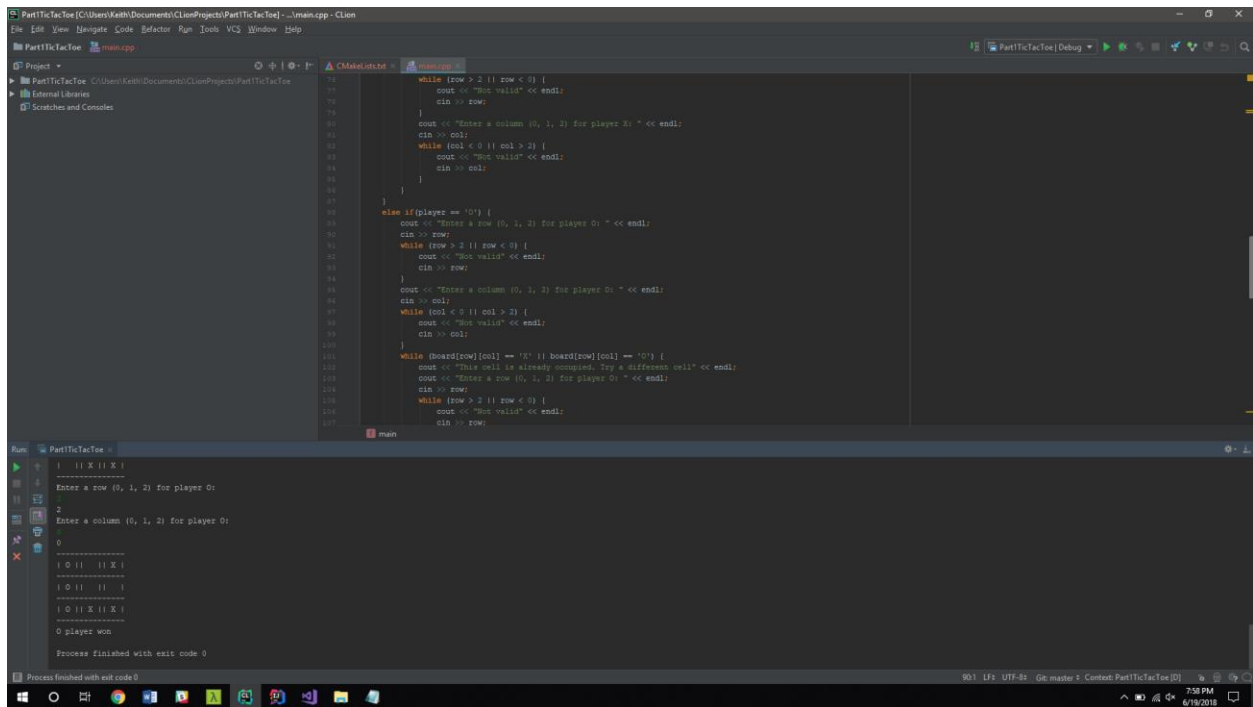
**Top Screenshot:** Shows the initial state of the game. The board is empty. Player X is prompted to enter a row and column. The code includes a while loop for row input and a while loop for column input. The board is displayed as a 3x3 grid of empty spaces.

```

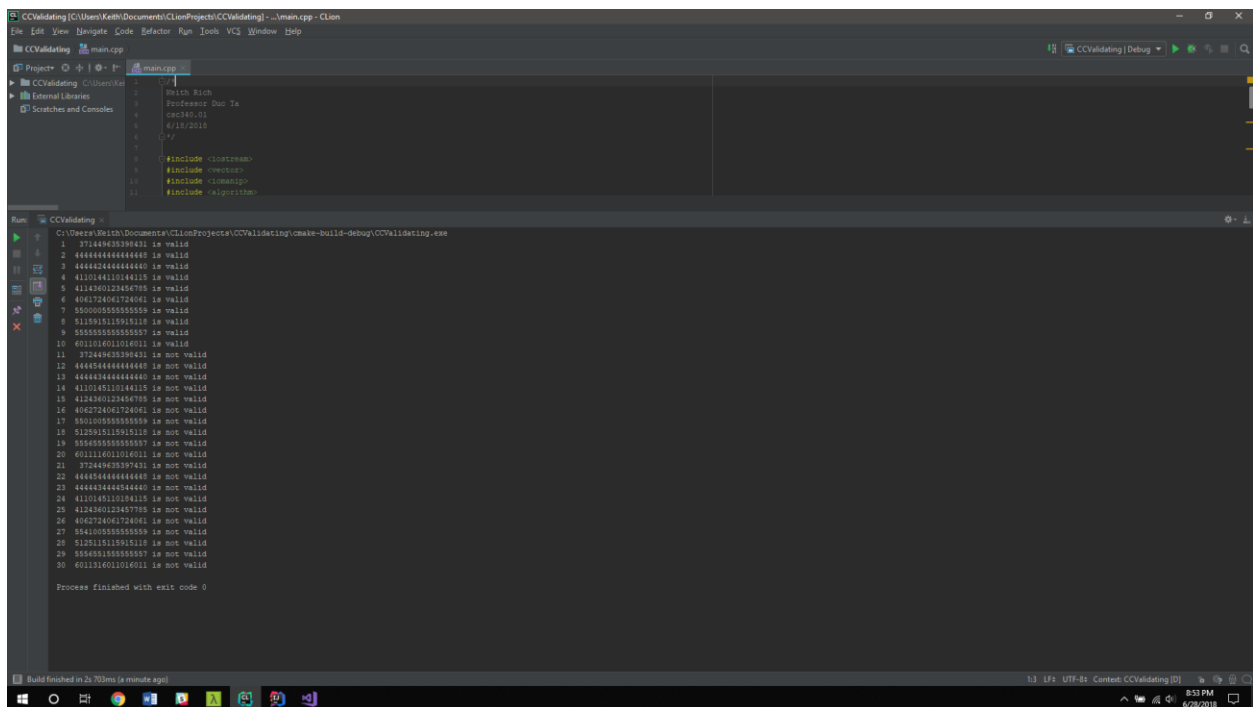
101 while (row > 2 || row < 0) {
102     cout << "Not valid" << endl;
103     cin >> row;
104 }
105 cout << "Enter a column (0, 1, 2) for player X: " << endl;
106 cin >> col;
107 while (col < 0 || col > 2) {
108     cout << "Not valid" << endl;
109     cin >> col;
110 }
111 }
112 else if (player == 'O') {
113     cout << "Enter a row (0, 1, 2) for player O: " << endl;
114     cin >> row;
115     while (row > 2 || row < 0) {
116         cout << "Not valid" << endl;
117         cin >> row;
118     }
119     cout << "Enter a column (0, 1, 2) for player O: " << endl;
120     cin >> col;
121     while (col < 0 || col > 2) {
122         cout << "Not valid" << endl;
123         cin >> col;
124     }
125     while (board[row][col] == 'X' || board[row][col] == 'O') {
126         cout << "This cell is already occupied. Try a different cell" << endl;
127         cout << "Enter a row (0, 1, 2) for player O: " << endl;
128         cin >> row;
129         while (row > 2 || row < 0) {
130             cout << "Not valid" << endl;
131             cin >> row;
132         }
133     }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

**Bottom Screenshot:** Shows the game state after several moves. The board is now partially filled with X and O. Player O is prompted to enter a row and column. The code includes a while loop for row input and a while loop for column input. The board is displayed as a 3x3 grid with X and O characters. The game logic includes checking for a win, a draw, or a valid move. The output shows "No winner" and "Process finished with exit code 0".



## -Part2



## -Part3

```
ReadFileDictionary [C:\Users\Kath\Documents\ClionProjects\ReadFileDictionary\...main.cpp - Clion
File Edit View Navigate Code Refactor Run Tools VCS Window Help

ReadFileDictionary main.cpp
Project ReadFileDictionary Clion
External Libraries
Scratches and Consoles

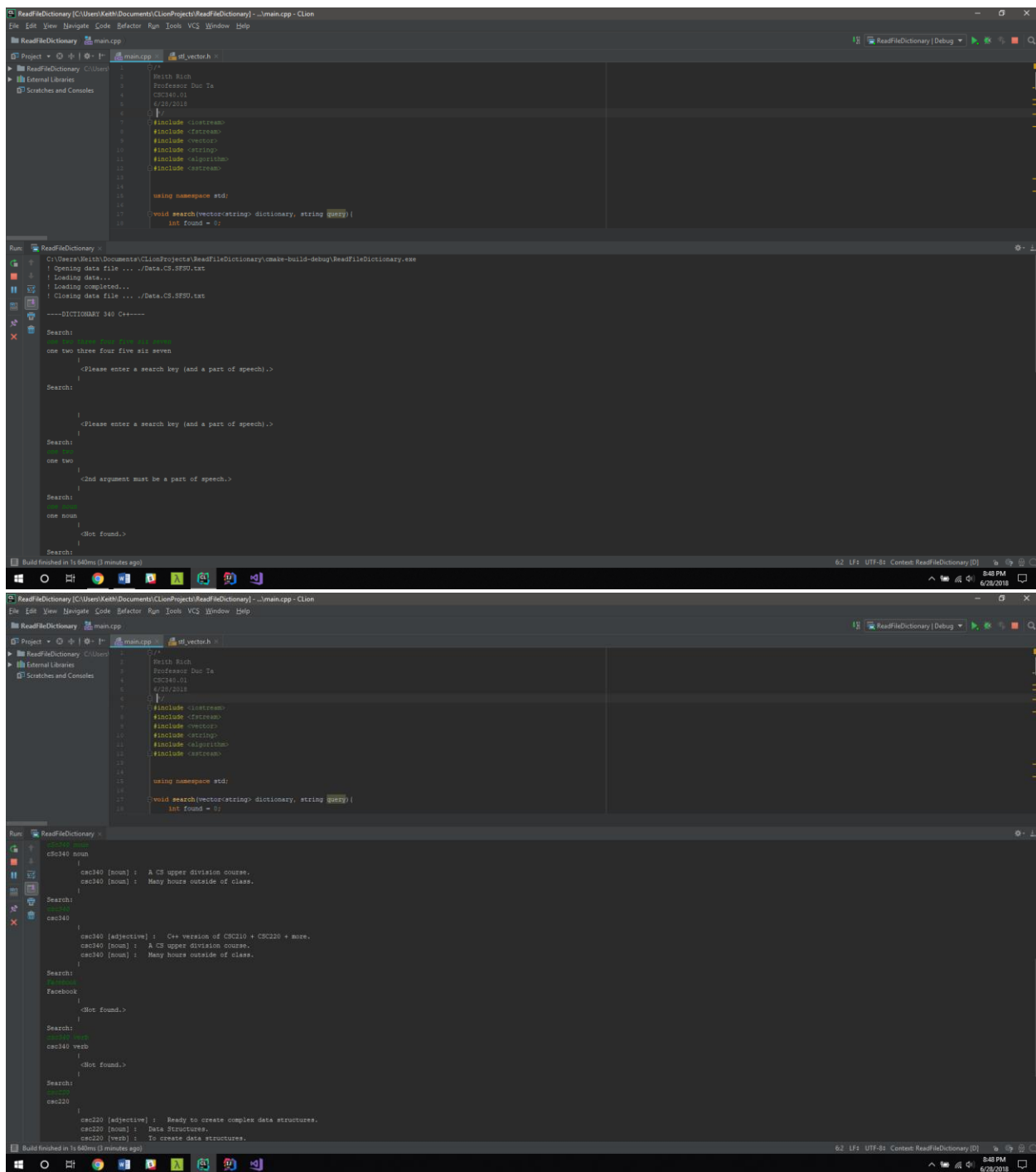
1 //
2 Keith Rich
3 Professor Duu Ta
4 CSC240.01
5 6/25/2018
6 //
7 #include <iostream>
8 #include <fstream>
9 #include <vector>
10 #include <string>
11 #include <algorithm>
12 #include <sstream>
13
14 using namespace std;
15
16 void search(vector<string> dictionary, string query) {
17     int found = 0;
18 }
```

Run: ReadFileDictionary

```
csc240 web
|
| <Not found.>
|
| Search:
|
| csc220
|
| csc220 [adjective] : Ready to create complex data structures.
| csc220 [noun] : Data Structures.
| csc220 [verb] : To create data structures.
|
| Search:
|
| csc220 web
|
| csc220 [verb] : To create data structures.
|
| Search:
|
| csc210
|
| csc210 [adjective] : Comfortable With Objects and Classes.
| csc210 [adjective] : Ready for CSC 220.
| csc210 [noun] : Intro to Java.
| csc210 [verb] : To learn Java.
|
| Search:
|
| IQ
|
| Process finished with exit code 0
```

88.1 LF: UTF-8: Content: ReadFileDictionary.ID

8:48 PM  
6/26/2018



4. In at least half a page, please explain clearly:

a. Your approach in implementing your program.

Most of my time went to trying to make sure that everything was formatted in identical way. I know I needed a search function algo as well as a function that formatted the given text file. The next challenge came from taking using input and making sure it catches the correct values. There was also the challenge of having to display more than one definition with only a single word.

b. Which data structures did you use?

I used a vector for the entire thing. One vector to store the lines of text from the original file. Another vector to format that vector. And lastly another vector that had the user input.

c. What did you use the data structures for?

I used the user input vector to know if I had the correct amount of words required and could display the correct definition(s). I used a formatted vector to store the items and their corresponding definitions be. This is because it is easy to iterate though a formatted vector and cross check their values.

d. Why these data structures but not the others?

I am most familiar with vectors and how to iterate though them.