Project 1

<Wordle Game>

Cis-17C

Name: Anangafac Alabaweh

Date: 05/08/22

Profressor: Mark Lehr

Introduction

I was drawn to the wordle game because it is a great way to test one's Primary English understanding and vocabulary. Furthermore, I usually experience word blocks in conversations, and Wordle has been an excellent resource for expanding my vocabulary. Moreover, Wordle is a simple game that tests one's ability to think critically.

The game consisted of 838 lines, three different difficulty levels, a class, four queues, two stacks, and five lists. I spent sixteen days coding it. I used the wordle game from the new york times website and mikhad.github.io/wordle/ GitHub repository as references.

Approached to Development

When I wrote the first engine for the game, it had a lot of errors; it frequently crashed without reason and could not determine duplicate values.

The Approached used for the current version Is;

1) My first step was to create a new list called copy_p2list and copy the word entered by the player into this list.

First, I created a function (process1) for determining whether a character is in the correct location.

If there is a character in the correct position, I replace it with an uppercase 'T' and store the character's current position in a new list. The comparison was accomplished by using two queues.

2) After process1 is complete, the program proceeds to process 2. This is done by comparing the user's word with the current wordle word. Because I had already used process1 to determine if any

words were in the correct position. In the process2 function, the characters are compared, and if they are not the same, the character is searched using copy_p2list as the library. If the character is in the copy_p2list, it is replaced with zero, followed by replacing zero with 'F'. This helps determine and eliminate faulty outputs.

Additionally, if the character is not found in the copy_p2list list, it means the character is not part of the wordle word, so it is replaced with an asterisk "*", which indicates the character is not part of the wordle puzzle.

After the process1 and process2 functions have completed executing, the word entered by the user is saved first in queue_results, followed by the results from the copy_p2list list. It is copied to the queue_results and displayed to the player.

3) Word_gen: The wordle words are pre-defined in a map/set, the key is the word_size, and the container is a set of strings. In every new game, the word_size is passed into the word_gen function, which determines the appropriate key and set based on the game level. After selecting the right set, a random word is selected from the set and returned.

4) Palindrome_check: The palindrome check is an additional feature I added to the game; this function determines if the word entered by the player is a palindrome; I use the front of the stack and queue, compare them; If they are all the same, the word is a palindrome;

5) display_queue: this function displays the word entered by the player and also the results after going through process1 and process2.

6) The four_word, five_word, and six_word functions are for each level. There are four-word, five-word, and six-word levels. Depending on the level selected by the player, the user is prompted to enter the number of letters of words. If the player's number of letters is less or more than the required amount,

the game will keep prompting the player to enter the correct number of letter words. When the player enters a word with the correct number of letters, it is copied to mylist character by character. This is then passed to both processes 1 and 2 for processing; the result of the game is then copied from mylist to queue_result using a range base for loop;

Game Rules:

- Depending on the level of the game, the user has to enter the required number of words; if the user enters less or more, they won't be able to go to the next stage of the play.
- The user has only 6 attempts to determine the wordle word
- The player has to enter only words in the English dictionary

Results Display:

T= The character is in the wordle word and also in the correct position

F= The Character is in the wordle word but in the wrong position

*****= The Character is not in the wordle word

If the player gets all T, for example, **"T T T T T,"** the player won. If there is any "**F**" in the result, the player didn't win

Type	Variable Name	Description	Location

Iterator	it	Random-Access iterator	Process1
	*it, *it1	Insert iterator	Process1
	it2	Random-Access iterator	
	*it2	Insert iterator	process2
	it3	Bidirectional iterator	process2
	it_toli++	forward iterator	display results
	mylist_it	Random-Access iterator	fivewords, fourwords, sixwords
	count_it	forward iterator	fivewords, fourwords, sixwords
list	copy_p2list	container that holds char values	process2
	mylist1	container that holds char values and the wordle word	four_words, five_words, six_words
	mylist	list container that holds the char characters of the word entered by the user	Main, process1, process2, fivewords, fourwords, sixword
	tocheli	integer list container that holds the position of every character	Main, process1, process2, fivewords, fourwords, sixword
	results		
stack	stk	stack/list container used to compare if the word entered by the user us a palidrome	Main, palidrome_check
Queue	que	queue/list of char used to compare if the word entered by the use is palidrome	Main, palidrome_check
	que, que1	use to compare the characters from mylist, and mylist1	Main, process1, process2
	queue_result	queue/list char container holds the results and word after every entry	Main
Map	words	Map/set associative container holds the wordle words for each level	word_gen
Set	words	Map/set associative container holds the wordle words for each level	word_gen
Algorithm	Advance	used to increment the iterator so it randomly selects a word from the set	word_gen
	find	used to the search if a character is part of the Wordle	process1, process2,
	count	used to count the occurrence of T, if the player got the right word	main, process1, process2, fivewords, fourwords, sixword
	sort	used to sort the Map/set-associative container	word_gen
	swap	used the clear the queue in the stack after every level play	clear
integer	trail_count	hold the number of times the player has played	main

	mylist_ct	hold the number of times the player has played	Main, process1, process2, fivewords, fourwords, sixword
	counter	hold the number of times the player has played	Main, process1, process2, fivewords, fourwords, sixword
	n1	variable passed in the display_queue function	display_queue
string	name	use to hold the words entered by the player	Main
	line	use to the hold the words read the rules files	Main
	word	use to hold the random word generated by the word_gen function	fivewords, fourwords, sixword
Char	m1, m2	use to hold the characters from the que and que1 before they are popped	process1, process2,
	st,qu	use to hold the characters from the que and que1 before they are popped	Palidrome_check
	Welcome_option	use to hold the player choice from the sub menu	Main
	play_option	use to hold the player choice from the main menu	Main
	quit_option, quit_option2	hold the player quit the choice option	

Pseudocode:

Process1

```
Start
While loop start
    Copy the characters with the wordle word characters
    If the are the same, replace with T
    Pop the queues
While loop end;
```

Process2:

```
Start
Copy the results from process1 to a new list
```

```
Compare the characters, if they are not the same,
Search for the character in the p1list;
```

Clear:

```
Start
Create a new queue and swap it with the old queue;
End
```

Palidrome_check:

```
If found, replace with F
Else replace with *
End;
```

```
Start
Compare the word enter by the use
Using the stack and queue
If the stack is equal to the queue
The word is a palindrome
else the word is  not a palindrome
end
```

Word_gen:

```
Start
Randomly select words from the set depending on the key
Return the  word;
End
```

Four_words:

```
Start
Prompt player to enter 4 letter word
Convert the word to characters and copy it to mylist
Call process1 and process2
Call the display function;
```

```
Clear all the containers
```

```
End;
```

Five_words:

```
Start
Prompt player to enter 5 letter word
Convert the word to characters and copy it to mylist
Call process1 and process2
Call the display function;
Clear all the containers
End;
```

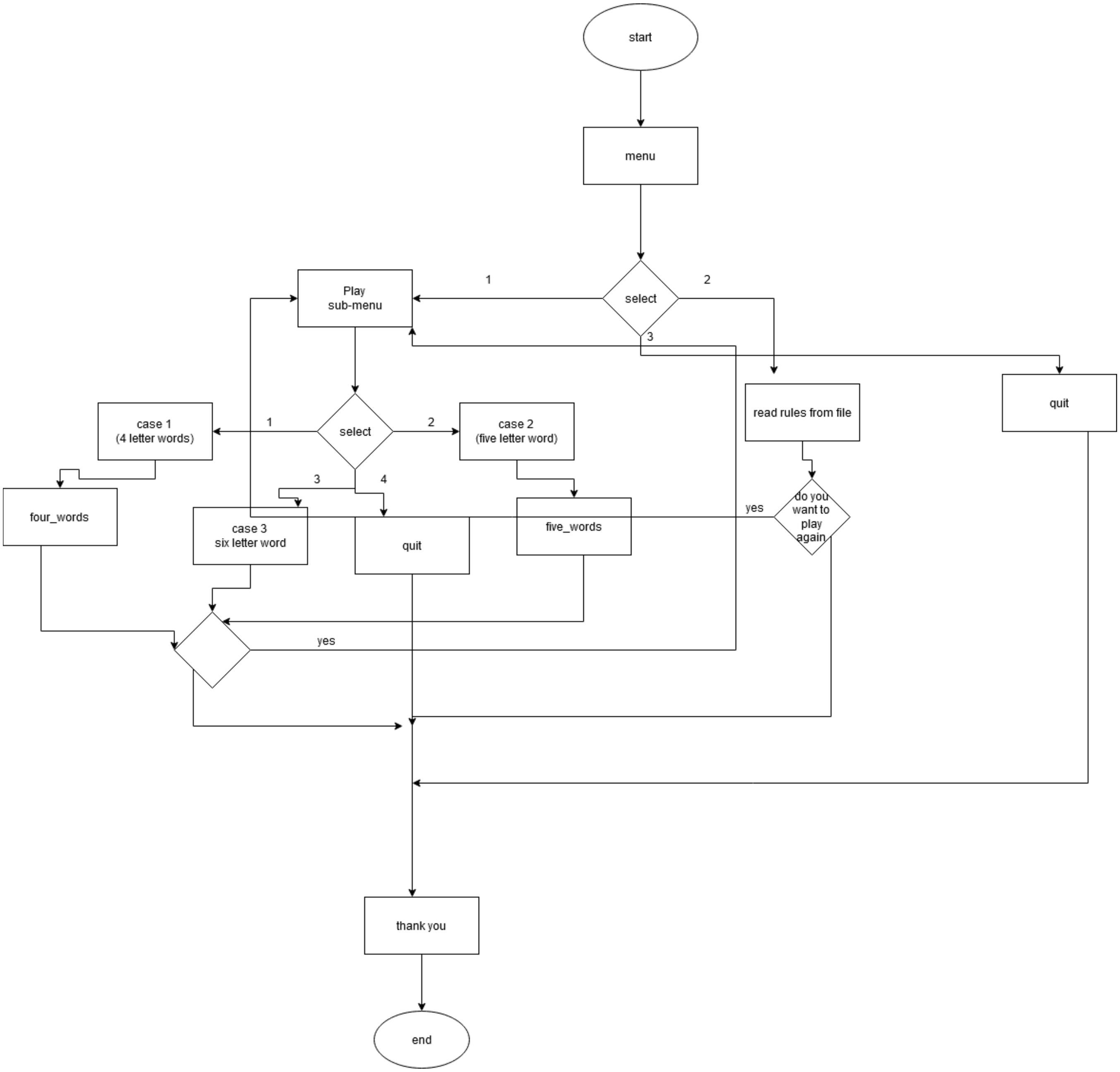
six_words:

```
Start
Prompt player to enter 6 letter word
Convert the word to characters and copy it to mylist
Call process1 and process2
Call the display function;
Clear all the containers
End;
```

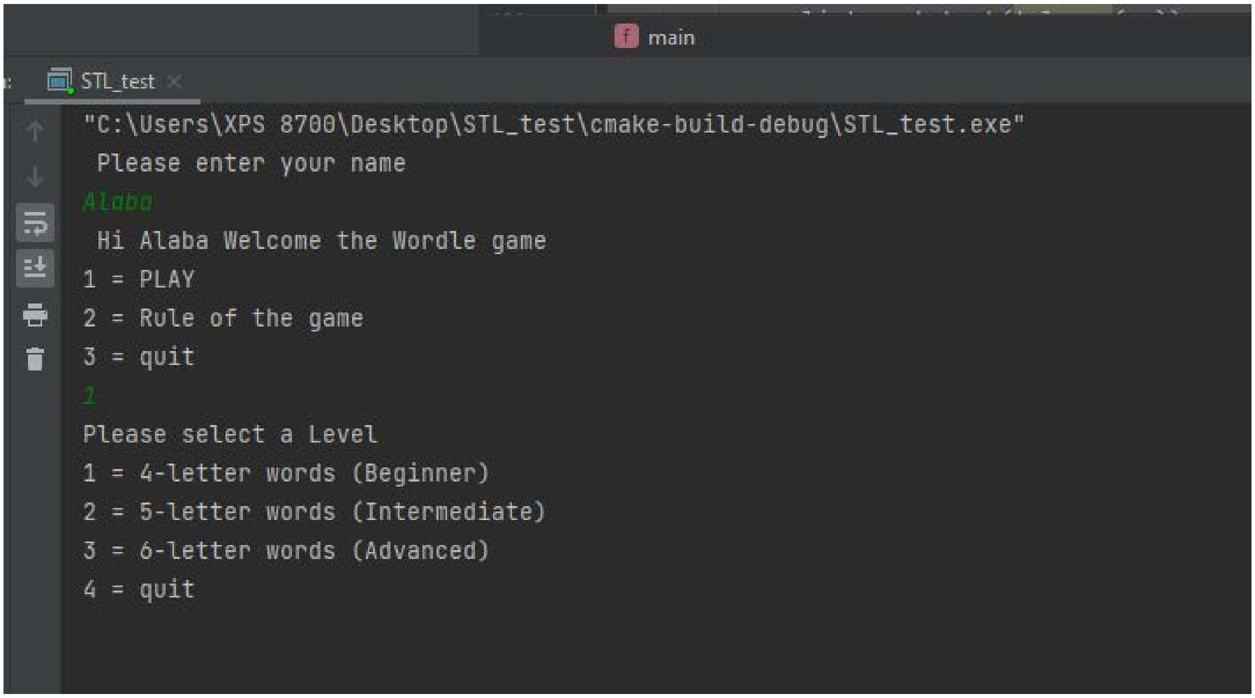
Main:

```
Start
Prompt user to select from menu
If the player selects 1:
Prompt the player to select the level they want to play
If 1: start the four-word function
If 2: start the five-word function

If 3: start the six-word function

If 4: quit the game:

If the player select 2:

Display the rules of the game

If the player selects 3:

Quit

End
```

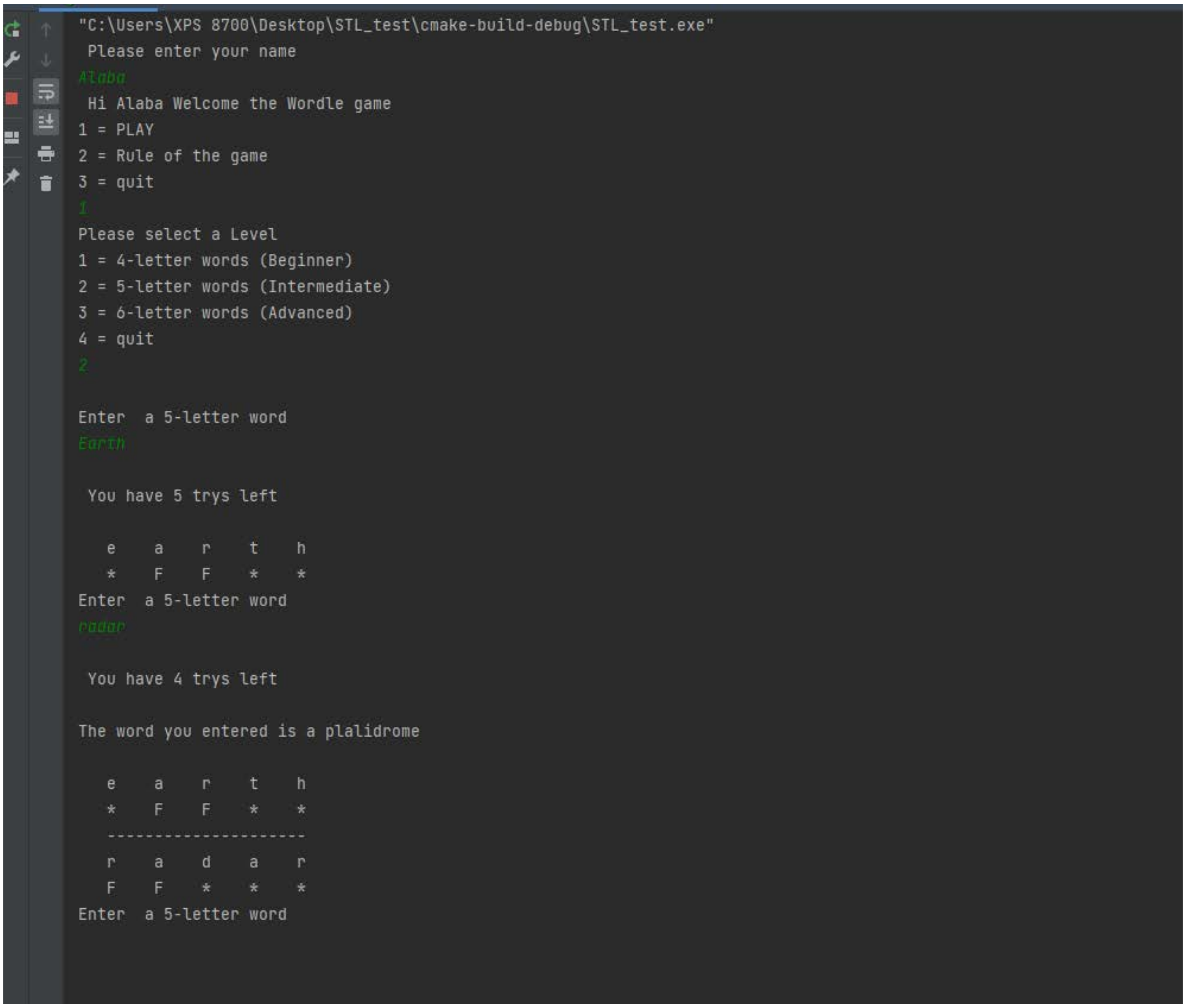
Sample Input/ output



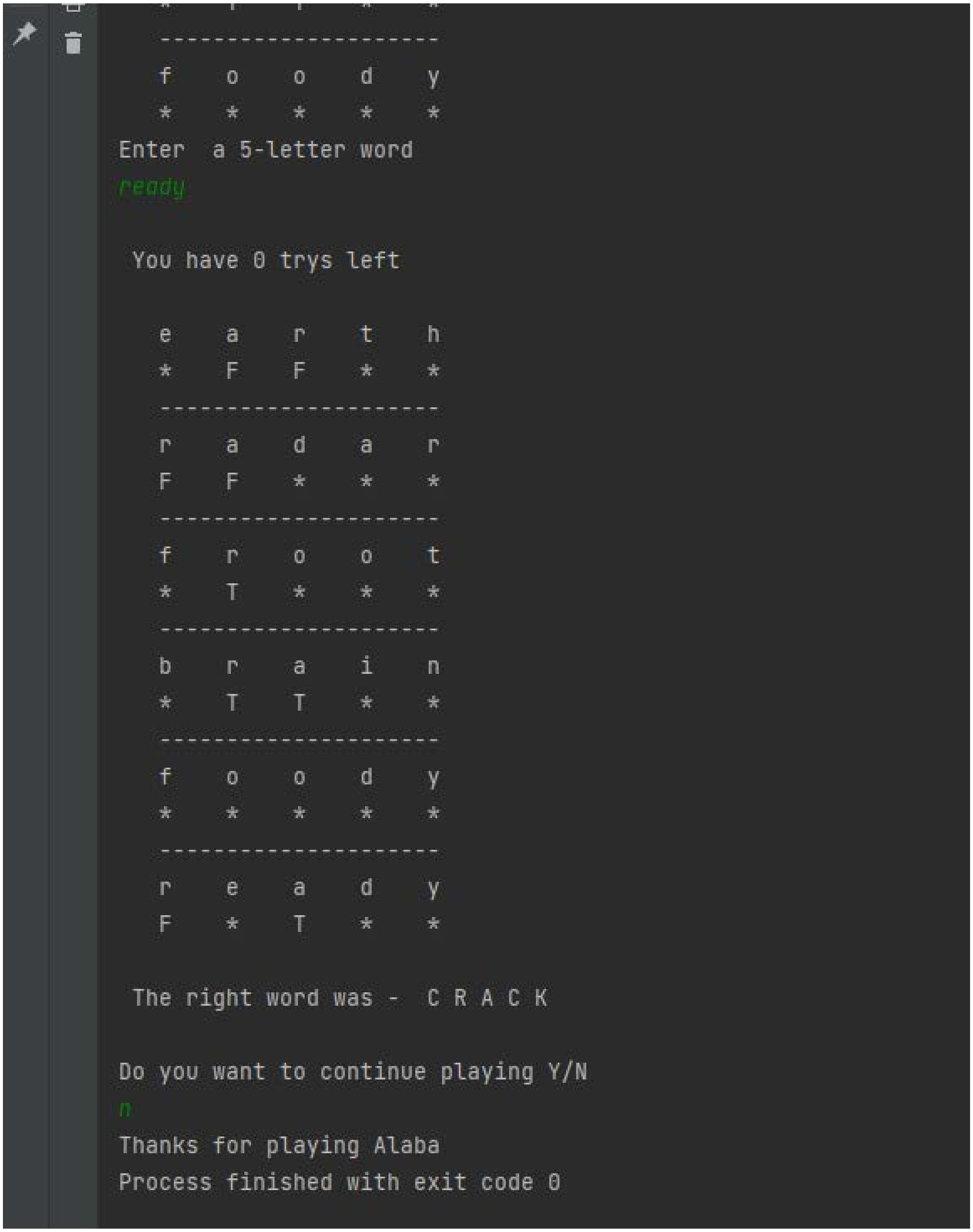
Sample input/output



```
"C:\Users\XPS 8700\Desktop\STL_test\cmake-build-debug\STL_test.exe"
 Please enter your name
Alaba
 Hi Alaba Welcome the Wordle game
1 = PLAY
2 = Rule of the game
3 = quit
1
Please select a Level
1 = 4-letter words (Beginner)
2 = 5-letter words (Intermediate)
3 = 6-letter words (Advanced)
4 = quit
```

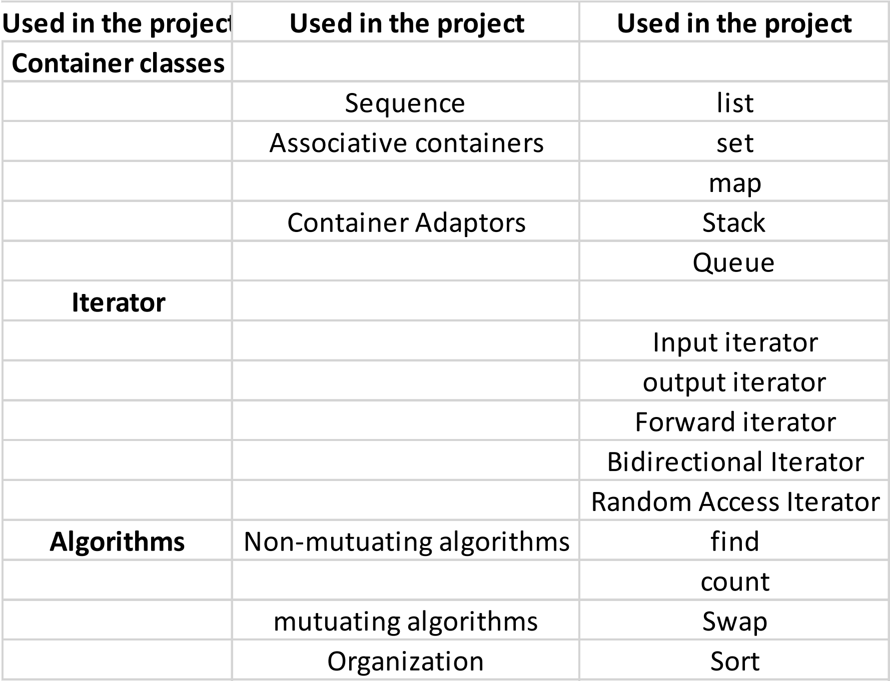


```
"C:\Users\XPS 8700\Desktop\STL_test\cmake-build-debug\STL_test.exe"
 Please enter your name
Alaba
 Hi Alaba Welcome the Wordle game
1 = PLAY
2 = Rule of the game
3 = quit
1
Please select a Level
1 = 4-letter words (Beginner)
2 = 5-letter words (Intermediate)
3 = 6-letter words (Advanced)
4 = quit
2

Enter  a 5-letter word
Earth

 You have 5 trys left

   e    a    r    t    h
   *    F    F    *    *
Enter  a 5-letter word
radar

 You have 4 trys left

The word you entered is a plalidrome

   e    a    r    t    h
   *    F    F    *    *
   ---------------------
   r    a    d    a    r
   F    F    *    *    *
Enter  a 5-letter word
```



```
   ---------------------
   f    o    o    d    y
   *    *    *    *    *
 Enter  a 5-letter word
 ready

  You have 0 trys left


   e    a    r    t    h
   *    F    F    *    *
   ---------------------
   r    a    d    a    r
   F    F    *    *    *
   ---------------------
   f    r    o    o    t
   *    T    *    *    *
   ---------------------
   b    r    a    i    n
   *    T    T    *    *
   ---------------------
   f    o    o    d    y
   *    *    *    *    *
   ---------------------
   r    e    a    d    y
   F    *    T    *    *

  The right word was -  C R A C K

 Do you want to continue playing Y/N
 n
 Thanks for playing Alaba
 Process finished with exit code 0
```

Checkoff Sheet Contents



Used in the projec	Used in the project	Used in the project
Container classes		
	Sequence	list
	Associative containers	set
		map
	Container Adaptors	Stack
		Queue
Iterator		
		Input iterator
		output iterator
		Forward iterator
		Bidirectional Iterator
		Random Access Iterator
Algorithms	Non-mutuating algorithms	find
		count
	mutuating algorithms	Swap
	Organization	Sort

Reference:

The C++ Standard Library: Second edition, Nicolai M Josuttis

C++ From control structures through Objects; Ninth edition, Tonny Gaddis

Wordle, New your times; https://www.nytimes.com/games/wordle/index.html

GeeksforGeeks; https://www.geeksforgeeks.org