CAB201 Programming Principles - Semester 2, 2019 Report for Assignment: Project – Genomic Sequence Retrieval - Part II

Student name and number:

[Enter student name and student number here, e.g. John Smith n9999999]

Build and Run Instructions

Please provide clear step-by-step instructions here on how to build your program in Visual Studio and run your program in the command line, given your submitted zip folder. For each step, you should include a screenshot. You may expand the box if needed.

[Enter build & run instructions. Include where to find .sln file (with screenshot), how to build your program (with screenshots), how to find the .exe file (with screenshots), and finally a screenshot of your .exe program being run from the command line]

Make sure that these instructions are tested as follows:

1. Place your project (exactly what you are about to upload as a submission) on a USB disk
2. Log onto a QUT SEF computer lab – this must be a **new login session**
3. Follow your own build & run instruction
4. Test your program.

Note that this is what the markers will do – if your program fails to build and execute you will lose all marks for testing your code.

IF THE PROJECT DOES NOT BUILD & RUN YOU MUST DECLARE IT HERE

AS WELL AS IN THE STATEMENT OF COMPLETENESS

Statement of Completeness

This statement of completeness will need to *accurately* state the functionality which has been implemented. There will be a penalty of 3.5 marks (loss of 3.5 marks) for a non-completed or submitted statement of completeness, and a penalty of 1 mark for each inaccurate statement to a maximum of 3 marks.

In the following section, you are required to mark which functionality you have implemented. In the column on the right please mark 'Y' where you have completed this functionality, and 'N' where you have not. Please fill in any additional text boxes requested, and please note any limitations or bugs in the box at the end of each section. You may expand the table if you need more room for comments.

Basic Functionality		
Build & Run	When following the Build & Run instructions, the program successfully builds and runs. This was tested in a QUT SEF lab with a new login session, using the same zipped folder that is submitted.	Y
Basic itinerary output	The program displays the data from the file	Y
	The program displays the appropriate line	Y
	The correct amount of information is displayed, e.g. only the relevant entries	Y
	The correct level, provided as a command line flag ***-levelN***, is executed	Y
	The program **does not** store the whole file in memory, instead it accesses the file on disk	N
Error handling	A clear error message is displayed when an incorrect number of arguments is provided	Y
	A clear error message is provided when an incorrect flag is provided (e.g. not -level1, etc.)	Y
	A clear error message is provided when the input file doesn't exist, or is incorrectly formatted	Y
Comments	*IndexSequence is still a work in progress and so is not included in these tests*	

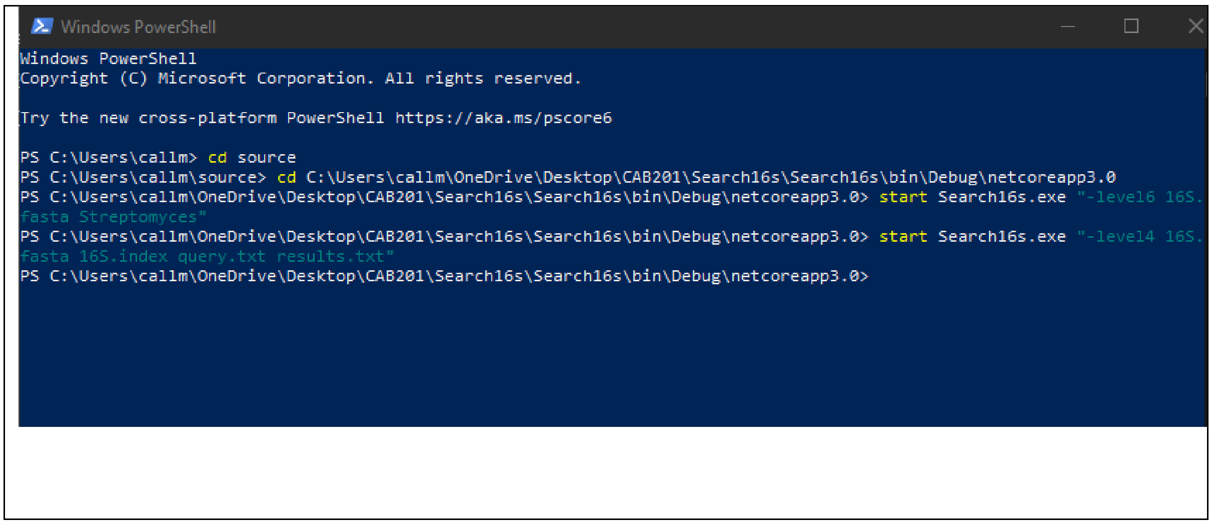
Searching Algorithm – Part II Please underline, circle or highlight the levels that were completed.	
Algorithm	**Level:** Level 4, level 5, level 6, level 7
	Bonus: Sequence matching using wildcards *If completed, please highlight level 7 above and discuss how you approached the problem and why you solved it the way you did.* ***I approached this task by researching as much about regex in C# as I could. Regex was my biggest weakness last semester and I wanted to make sure I had no troubles this time.***
	Level 7 Explanation *If you have completed the optional level 7, please include details on how you solved it and any resources you may have used.* ***I solved the task using the task by replacing every asterisk (*) in the search parameter with the regex "[ACGT]*" and using the Regex.Match function. This allowed me to find all possible sequences that contained any number of the letters A,C,G or T where the asterisks (*) were located. I specified those specific letters for the regex as they are the only letters used in the DNA sequences and so I am not using system resources looking for letters that will not appear.*** *Resource Used:* https://www.dotnetperls.com/regex
Comments	*Please note any limitations, bugs, logical errors, differences from provided examples and possible explanations, and/or run-time errors here*

Screenshots of Functionality

In the following section, you are required to provide screenshots that provide evidence of your program working with provided input. You must complete this section.

1) The `16S.fasta file` has been provided with this template. Download them and place them in the same folder as your .exe file. You may have extra files, e.g. a query file, in this folder, and your .exe may be named differently. This is fine.
2) Open the command prompt and go to the above folder. In the command line, type the name of the .exe file and copy and paste following arguments:

```
Search16s -level4 16S.fasta 16S.index query.txt results.txt
```

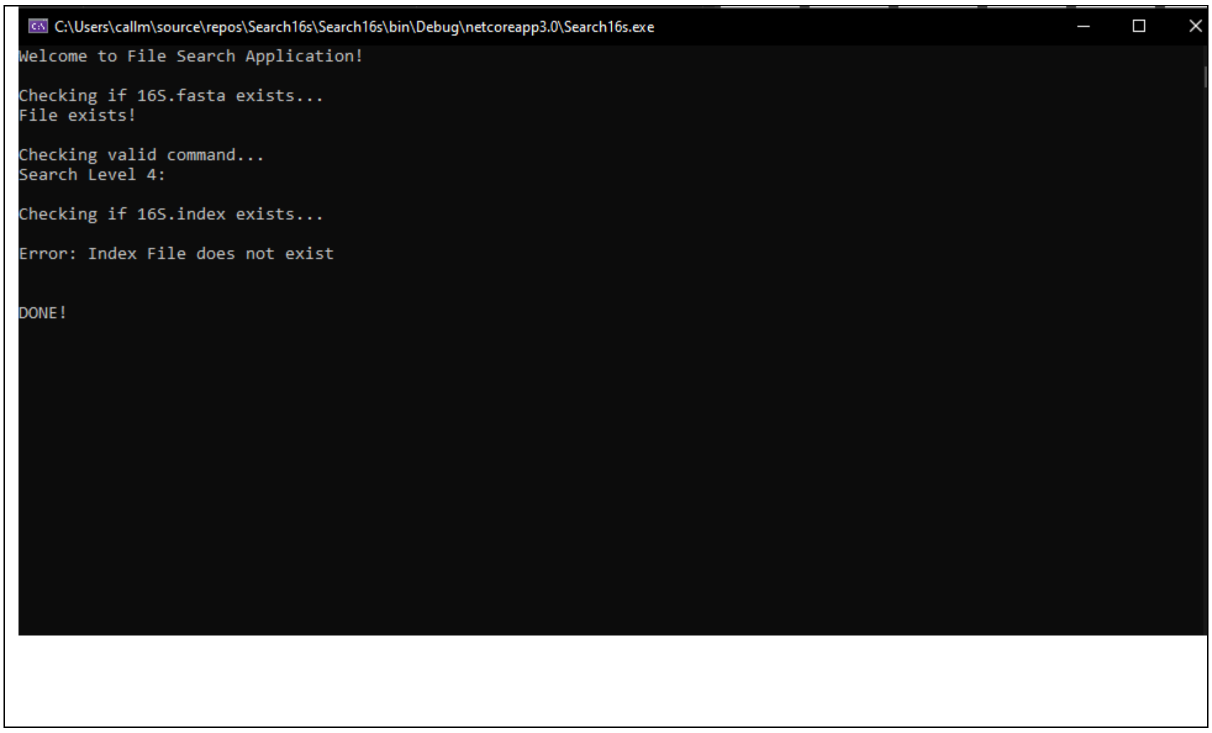


```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\callm> cd source
PS C:\Users\callm\source> cd C:\Users\callm\OneDrive\Desktop\CAB201\Search16s\Search16s\bin\Debug\netcoreapp3.0
PS C:\Users\callm\OneDrive\Desktop\CAB201\Search16s\Search16s\bin\Debug\netcoreapp3.0> start Search16s.exe "-level6 16S.
fasta Streptomyces"
PS C:\Users\callm\OneDrive\Desktop\CAB201\Search16s\Search16s\bin\Debug\netcoreapp3.0> start Search16s.exe "-level4 16S.
fasta 16S.index query.txt results.txt"
PS C:\Users\callm\OneDrive\Desktop\CAB201\Search16s\Search16s\bin\Debug\netcoreapp3.0>
```

3) Hit enter to run your program.



```
C:\Users\callm\source\repos\Search16s\Search16s\bin\Debug\netcoreapp3.0\Search16s.exe
Welcome to File Search Application!

Checking if 16S.fasta exists...
File exists!

Checking valid command...
Search Level 4:

Checking if 16S.index exists...

Error: Index File does not exist


DONE!
```

Self-Assessment:

1) How do I think I went with this assignment?

Insert Answer Here

2) What did I find difficult in this assignment?

Insert Answer Here

3) What would I do differently next time?

Insert Answer Here

4) Were there any bugs in my assignment, if so what were they?

Insert Answer Here

CRA:

Please fill out the following CRA, reporting how many marks you believe your project might be awarded. Your assessment should be a considered reflection on what you have achieved. The purpose of this is to advise the marker of what you believe was achieved in order for us to pay attention to discrepancies. **Your self-assessment is NOT attracting marks, but must be provided (penalty applies if missing)**.

<u>Code Quality</u> To score points in this section, the student must follow the code quality guidelines as specified in the C# Coding Style Guide on Blackboard	**/30**
Maintained consistent, clear, and meaningful standard in variable and method naming. No magic numbers.	/3
Well structured – consistent and appropriate white spacing, line length, indentation, and separation into files within the project (i.e. one class per file)	/2
Well commented – class header comment at beginning of each class, comment before every method, and in-line comments to explain complex or not easily discernible code. In-line comments are not excessive.	/4
The DRY principle (Don't repeat yourself) is followed where appropriate	/3
Methods are single purpose and clear	/4
Classes are well designed, with high cohesion and low coupling	/8
Classes are separated into reusable modules where appropriate	/3
Exceptions are thrown and handled appropriately	/3

<u>Basic Functionality</u> To score marks in this section, your program must be able to be run from the command line with the appropriate arguments.	**/15**

Basic Output	The program displays the data from the file	/1
	The program displays the appropriate line	/1
	The correct amount of information is displayed, e.g. only the relevant entries	/1
	The correct level, provided as a command line flag ***-levelN***, is executed	/1
	The program **does not** store the whole file in memory, instead it accesses the file on disk	/5
	Total:	**/9**
Error Handling	A clear error message is displayed when an incorrect number of arguments is provided	/2
	A clear error message is provided when an incorrect flag is provided (e.g. not -level1, etc.)	/2
	A clear error message is provided when the input file doesn't exist, or is incorrectly formatted	/2
	Total:	**/6**

<u>Part II</u> To score marks in this section, your program must be able to run levels 4-7.		**Marks Available:** **/55**
Level 4	The program creates a file as specified by the command line arguments	/2
	The index file contains a list of all the sequence ids with the appropriate byte-offset	/5
	The searching program makes use of the created index file to execute a number of queries	/5
	A clear error message is provided when the index file does not exist	/1

	Clear error messages are provided when the query file cannot be found, or when a bad query is given, like in Level 3	/2
	Total:	**/15**
Level 5	The program correctly locates and prints the requested sequence ids	/10
	A clear error message is provided when the sequence does not exist	/5
	Total:	**/15**
Level 6	The program correctly locates and prints the requested sequence ids	/15
	A clear error message is provided when the keyword does not exist	/5
	Total:	**/20**
Level 7 (Optional)	The program correctly decodes the expression given and identifies any matching sequences	/5
	A clear error message is provided when the sequence does not exist	/5
	Total:	**/+10**