

# **CAB201 Programming Principles - Semester 2, 2019**

## **Report for Assignment: Project – Genomic Sequence Retrieval - Part II**

**Student name and number:**

*[Phuong Nam Ly n10098097]*

### **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.

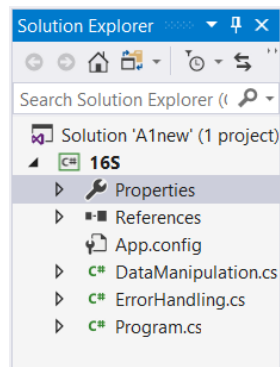
# Program User Manual

*Step-by-step instructions:*

*Decompress the A1new.zip file*

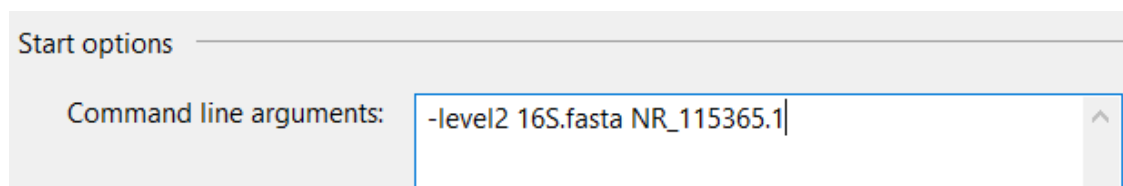
*Open the solution file A1new.sln, which is located inside of the decompressed A1 folder*

*Open the Properties, which is in the Solution Explorer*



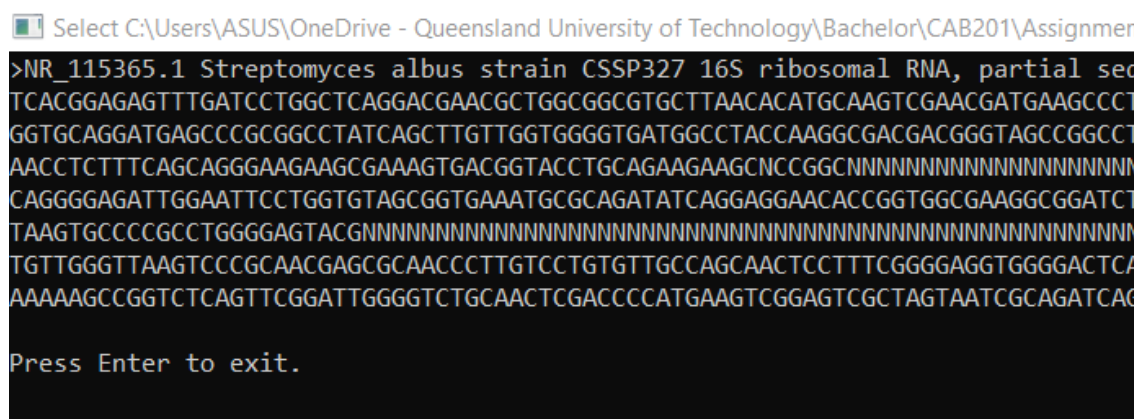
*In the “Debug” section, enter the command line.*

*A sample command line is entered.*



*Start the program in Debug mode by clicking the Start button (green) or use Ctrl + F5.*

*Wait for the output.*



**The program is applicable for Build and Run Instructions**

## 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		
<b>Build &amp; Run</b>	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/N
<b>Basic itinerary output</b>	The program displays the data from the file	Y/N
	The program displays the appropriate line	Y/N
	The correct amount of information is displayed, e.g. only the relevant entries	Y/N
	The correct level, provided as a command line flag - <b>levelN</b> , is executed	Y/N
	The program <b>does not</b> store the whole file in memory, instead it accesses the file on disk	Y/N
<b>Error handling</b>	A clear error message is displayed when an incorrect number of arguments is provided	Y/N
	A clear error message is provided when an incorrect flag is provided (e.g. not -level1, etc.)	Y/N
	A clear error message is provided when the input file doesn't exist, or is incorrectly formatted	Y/N
<b>Comments</b>	Only Level 1-4 has been implemented for the Progression Report Exception Handling has not been implemented so there might be bugs	

## Searching Algorithm – Part II

Please underline, circle or highlight the levels that were completed.

Algorithm	<b>Level:</b> Level 4, level 5, level 6, level 7
	<b>Bonus:</b> Sequence matching using wildcards  <i>If completed, please highlight level 7 above and discuss how you approached the problem and why you solved it the way you did.</i>
	<b>Level 7 Explanation</b> <i>If you have completed the optional level 7, please include details on how you solved it and any resources you may have used.</i>
Comments	Only Level 1-4 has been implemented for the Progression Report Exception Handling has not been implemented so there might be bugs

## Screenshots of Functionality

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.

**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
```

**Place screenshot of entered command line arguments here:**

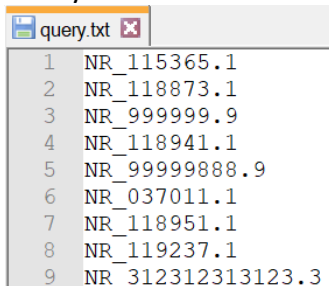
### 4.1 IndexedSequence

Start options

Command line arguments: 16S.fasta 16S.index

### 4.2 Successful search

Query.txt



```
1 NR_115365.1
2 NR_118873.1
3 NR_999999.9
4 NR_118941.1
5 NR_99999888.9
6 NR_037011.1
7 NR_118951.1
8 NR_119237.1
9 NR_312312313123.3
```

Start options

Command line arguments: -level4 16S.fasta 16S.index query.txt results.txt

### 4.3 Unsuccessful search

Query.txt

```
query.txt
1 NR_115365.1
2 NR_118873.1
3 NR_999999.9
4 NR_118941.1
5 NR_99999888.9
6 NR_037011.1
7 NR_118951.1
8 NR_119237.1
9 NR_312312313123.3

Start options

Command line arguments: -level4 16S.fasta 16S.index query.txt results.txt
```

3) Hit enter to run your program.

Place screenshot(s) of the full output to console of your program. You may expand the box as necessary, and use as many screenshots as needed:

#### 4.1 IndexedSequence

```
16S.index
1 NR_118889.1 0
2 NR_118899.1 1389
3 NR_074334.1 2841
4 NR_118873.1 2841
5 NR_119237.1 2841
6 NR_118890.1 4606
7 NR_044838.1 6033
8 NR_118908.1 7572
9 NR_118900.1 8995
10 NR_044822.1 10461
11 NR_026088.1 11809
12 NR_026093.1 13421
```

#### 4.2 Successful search

```
results.txt
1 NR_118873.1 Archaeoglobus fulgidus DSM 4304 strain VC-16 16S ribosomal RNA, com
2 ATTCTGGTTGATCCTGCCAGAGGCCGCTGCTATCCGGCTGGGACTAAGCCATGCGAGTCAAGGGGCTTGTATCCCTTCG
3 NR_119237.1 Archaeoglobus fulgidus DSM 4304 strain VC-16 16S ribosomal RNA, com
4 ATTCTGGTTGATCCTGCCAGAGGCCGCTGCTATCCGGCTGGGACTAAGCCATGCGAGTCAAGGGGCTTGTATCCCTTCG
5 NR_037011.1 Aeromonas salmonicida subsp. achromogenes strain 6263/4/5 16S ribos
6 GAGTTTGATCATGGCTCAGATTGAACGCTGGCGGCAGGCCTAACACATGCAAGTCGAGCGGCAGCGGGAAAGTAGCTTG
7 NR_118951.1 Bacillus azotoformans LMG 9581 strain ATCC 29788 16S ribosomal RNA,
8 GAGAGTTTANCCCTGGCTCAGGACGAACGCTGGCGGCGTGCCTNANACNTNCNAGTCGAGCGGATGATAAAGGAGCTTG
9 NR_118941.1 Streptococcus alactolyticus strain NCDO 1091 16S ribosomal RNA, par
10 GAACGGGTGAGTAACGCGTAGGTAACCTGCCTTGTAGCGGGGGATAACTATTGGAACGATAGCTAATACCGCATAACA
11 NR_115365.1 Streptomyces albus strain CSSP327 16S ribosomal RNA, partial sequen
12 TCACGGAGAGTTTGATCCTGGCTCAGGACGAACGCTGGCGGCGTGCTTAACACATGCAAGTCGAACGATGAAGCCCTTC
```

#### 4.3 Unsuccessful search

C:\Users\lyphu\OneDrive - Queensland University of Tech

```
Error, sequence NR_999999.9 not found
Error, sequence NR_99999888.9 not found
Error, sequence NR_312312313123.3 not found

Program finishes. Press Enter to exit.
```

## Self-Assessment:

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

I think I'm only 30% done overall as I only implement level 4.

*Limitations:*

*Only Level 1-4 has been implemented for the Progression Report*

*Class cohesion has not improved since Assessment 1*

*Exception Handling has not been implemented*

*Codes have not been commented*

*I still have to:*

- *Implement L5-6 as compulsory*
- *Improving class cohesion*
- *Exception Handling*
- *Comment the codes*
- *Implement L7 as optional*

2) What did I find difficult in this assignment?

*L4 is the most difficult so far as I need to understand about byte offset and byte size to implement the full functionalities for L4.*

3) What would I do differently next time?

*I would start sooner in order to finish L5-6 so that I can consider any possible bugs.*

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

*Exception Handling has not been implemented so there might be possible bugs*

*Level 1, if a letter is entered instead of a number for starting line or number of lines, there will be unhandled exceptions*

## 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).**

Code quality has not changed since Assignment 1 so self-assessment is zero.

<b><u>Code Quality</u></b>	
To score points in this section, the student must follow the code quality guidelines as specified in the C# Coding Style Guide on Blackboard	<b>0/30</b>
Maintained consistent, clear, and meaningful standard in variable and method naming. No magic numbers.	0/3
Well structured – consistent and appropriate white spacing, line length, indentation, and separation into files within the project (i.e. one class per file)	0/2
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.	0/4
The DRY principle (Don't repeat yourself) is followed where appropriate	0/3
Methods are single purpose and clear	0/4
Classes are well designed, with high cohesion and low coupling	0/8
Classes are separated into reusable modules where appropriate	0/3
Exceptions are thrown and handled appropriately	0/3

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



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

<b><u>Part II</u></b>		<b>Marks Available:</b>
To score marks in this section, your program must be able to run levels 4-7.		<b>12/55</b>
<i>Level 4</i>	The program creates a file as specified by the command line arguments	2/2
	The index file contains a list of all the sequence ids with the appropriate byte-offset	5/5
	The searching program makes use of the created index file to execute a number of queries	5/5
	A clear error message is provided when the index file does not exist	0/1

	Clear error messages are provided when the query file cannot be found, or when a bad query is given, like in Level 3	0/2
	<b>Total:</b>	<b>12/15</b>
<i>Level 5</i>	The program correctly locates and prints the requested sequence ids	/10
	A clear error message is provided when the sequence does not exist	/5
	<b>Total:</b>	<b>0/15</b>
<i>Level 6</i>	The program correctly locates and prints the requested sequence ids	/15
	A clear error message is provided when the keyword does not exist	/5
	<b>Total:</b>	<b>0/20</b>
<i>Level 7 (Optional)</i>	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
	<b>Total:</b>	<b>0/+10</b>