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

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

**Student name and number:**

*[Shu Du n10505024]*

**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**

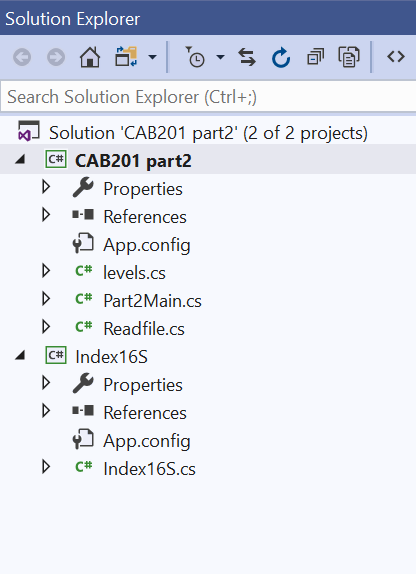
This statement of completeness will need to *accurately* state the functionality which has been implemented. There will be a penalty of 3.5 marks (loss of 3.5 marks) for a non-completed or submitted statement of completeness, and a penalty of 1 mark for each inaccurate statement to a maximum of 3 marks.

**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/N |
| **Basic itinerary output** | 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 ***-levelN***, is executed | Y/N |
| The program **does not** store the whole file in memory, instead it accesses the file on disk | Y/N |
| **Error handling** | 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 |
| **Comments** |  | |

|  |  |
| --- | --- |
| **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.* |
| **Level 7 Explanation**  *I used “Regex.Match” method after converting "\*" to "[A-Z]\*", then used the same search method from levels above.* |
| **Comments** |  |

Below is the layout of the project. To be able to run level4, user needs to run “Index16S” to generate 16S index file first.



**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

|  |
| --- |
| **Screenshots of command line and results:**  **Level 4.1-indexfile**      **Level 4.2:**    **Level5:** |

|  |
| --- |
| **Level6:**    **Level7:** |

**Self-Assessment:**

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

**I think I went ok on the part 2 as I made a lot of improvement based on the part1 submission’s feedback. I have simplified the whole project and add more exception handling than the last one.**

2) What did I find difficult in this assignment?

3) What would I do differently next time?

**How to organize the whole project in a good OOP manner is difficult. Also, the error handling.**

**If I am doing a larger project next time, I will try to use more techniques in OOP such as abstract class, polymorphism in order to get the code more organized.**

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

**I don’t see any bugs so far.**

**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**  To score points in this section, the student must follow the code quality guidelines as specified in the C# Coding Style Guide on Blackboard | **26/30** |
| Maintained consistent, clear, and meaningful standard in variable and method naming. No magic numbers. | 3/3 |
| Well structured – consistent and appropriate white spacing, line length, indentation, and separation into files within the project (i.e. one class per file) | 2/2 |
| 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. | 3/4 |
| The DRY principle (Don’t repeat yourself) is followed where appropriate | 3/3 |
| Methods are single purpose and clear | 4/4 |
| Classes are well designed, with high cohesion and low coupling | 6/8 |
| Classes are separated into reusable modules where appropriate | 3/3 |
| Exceptions are thrown and handled appropriately | 2/3 |

|  |  |  |
| --- | --- | --- |
| Basic Functionality  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 | 1/1 |
| The program displays the appropriate line | 11/1 |
| The correct amount of information is displayed, e.g. only the relevant entries | 1/1 |
| The correct level, provided as a command line flag ***-levelN***, is executed | 1/1 |
| The program **does not** store the whole file in memory, instead it accesses the file on disk | 5/5 |
| **Total:** | **9/9** |
| *Error Handling* | 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 |
| **Total:** | **6/6** |

|  |  |  |
| --- | --- | --- |
| Part II  To score marks in this section, your program must be able to run levels 4-7. | | Marks Available: |
| 55/55 |
| *Level 4* | 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 | 1/1 |
| Clear error messages are provided when the query file cannot be found, or when a bad query is given, like in Level 3 | 2/2 |
| **Total:** | **15/15** |
| *Level 5* | The program correctly locates and prints the requested sequence ids | 10/10 |
| A clear error message is provided when the sequence does not exist | 5/5 |
| **Total:** | **15/15** |
| *Level 6* | The program correctly locates and prints the requested sequence ids | 15/15 |
| A clear error message is provided when the keyword does not exist | 5/5 |
| **Total:** | **20/20** |
| *Level 7 (Optional)* | The program correctly decodes the expression given and identifies any matching sequences | 5/5 |
| A clear error message is provided when the sequence does not exist | 5/5 |
| **Total:** | **+10/+10** |