COMP9319 2017s1 Assignment 2: Searching BWT Encoded File

Your task in this assignment is to create a search program that implements BWT backward search, which can efficiently search a BWT encoded record file. The original file (before BWT) format is:

```
[<recordID1>]<text1>[<recordID2>]<text2>[<recordID3>]<text3>... ...
```

where `<recordID1>`, `<recordID2>`, `<recordID3>`, etc. are integer values that are used as unique identifiers;
and `<text1>`, `<text2>`, `<text3>`, etc. are text values, which include any visible ASCII alphabets (i.e., any character with ASCII value from 32 to 126), tab (ASCII 9) and newline (ASCII 10 and 13). For simplicity, there will be no open or close square bracket in the text values.

Your C/C++ program, called **bwtsearch**, accepts the path to a BWT encoded file; the path to an index file; and **one** to **three** quoted query strings (i.e., search terms) as commandline input arguments. Each search term can be up to 256 characters. Using the given query strings, it will perform backward search on the given BWT encoded file, and output all the records that contain ALL input query strings (i.e., a boolean AND search) to the standard output.

To make the assignment easier, we assume that: (1) the search is case sensitive; (2) the output records do not need to be sorted; (3) duplicates are allowed if multiple matches are found in a record. Your output shall include the identifiers as well as the text values, one line (ending with a '\n') for each match. If there are newlines in a text value, output them as is, and then ending the record with a '\n'. If there are multiple matches in one record, that record will be output multiple number of times (same number as the number of matches in that record, see an example below).

Your solution is allowed to write out **one** external index file that is no larger than the size of the given, input BWT file. If your index file is larger than the size of the input BWT file, you will receive zero points for the tests that using that file. You may assume that the index file will not be deleted during all the tests for a given BWT file, and all the test BWT files are uniquely named. Therefore, to save time, you only need to generate the index file when it does not exist yet.

Although you do not need to submit a BWT encoder, it is a part of this assignment that you will implement a simple BWT encoding program (this will help you in understanding the lecture materials and assist in testing your assignment).

Example

Suppose the original file (dummy.txt) before BWT is:

```
[3]Computers in industry[25]Data compression[33]Integration[40]Big data indexing
```

(Note that you will not be given the original file. You will only be provided with the BWT encoded file.)
Given the command:

```
%wagner> bwtsearch ~MyAccount/XYZ/dummy.bwt dummy.idx "in"
```

The output should be:

```
[3]Computers in industry
[3]Computers in industry
[40]Big data indexing
[40]Big data indexing
```

Note that the order of the records can be different and the output is still correct, e.g.,

```
[40]Big data indexing
[3]Computers in industry
[3]Computers in industry
[40]Big data indexing
```

Another example:

```
%wagner> bwtsearch ~MyAccount/XYZ/dummy.bwt dummy.idx "in "
```

The output should be:

```
[3]Computers in industry
```

And the last example:

```
%wagner> bwtsearch ~MyAccount/XYZ/dummy.bwt dummy.idx " in" "ata"
```

The output should be:

```
[40]Big data indexing
```

You can find some more sample bwt files by logging into CSE machines and going to folder ~cs9319/a2. Again, note that the original text files (i.e., .txt files) are provided there for your reference only. Your solution should not assume the availability of the original text files during the assignment marking.

We will use the `make` command below to compile your solution. Please provide a makefile and ensure that the code you submit can be compiled. Solutions that have compilation errors will receive zero points for the entire assignment.

```
    make
```

Your solution will be compiled and run on a typical CSE Linux machine e.g. wagner. Your solution should **not** write out any external files other than the index file. Any solution that writes out external files other than the index file during any time of their program execution will receive zero points for the entire assignment.

Performance

Your solution will be marked based on space and runtime performance. Your soluton will not be tested against any BWT encoded files that are larger than 160MB.

Runtime memory is assumed to be always less than 10MB. Runtime memory consumption will be measured by `valgrind massif` with the option `--pages-as-heap=yes`, i.e., all the memory used by your program will be measured. Any solution that violates this memory requirement will receive zero points for that query test. Any solution that runs for more than **120 seconds** on a machine with similar specification as *wagner* for the first query on a given BWT file will be killed, and will receive zero points for the queries for that BWT file. After that any solution that runs for more than **30 seconds** for any one of the subsequent queries on that BWT file will be killed, and will receive zero points for that query test.

We will use the `time` command and count both the user and system time as runtime measurement.

Documentation

You will be marked on your documentation of your comments for variables, functions and steps in your solution. Your source code will be inspected and marked based on readability and ease of understanding.

Assumptions/clarifications/hints

1. To avoid long output time, none of the testcases for marking will result in outputting more than 500 records.
2. To avoid large runtime memory for sorting, none of the testcases for marking will result in more than 5,000 matches for each search term.
3. The input filename is a path to the given BWT encoded file. Please open the file as read-only in case you do not have the write permission.
4. Marks will be deducted for output of any extra text, other than the required, correct answers (in the right order). This extra information includes (but not limited to) debugging messages, line numbers and so on.
5. You can assume that the input query string will not be an empty string (i.e., "").
6. If a search term is a number (e.g., "30"), it shall not match any identifiers that contain the number (e.g., [130]); and it shall only match the text values that include the number (e.g., [17] I have 30 apples in my bag).
7. You are allowed to use one external index file to enhance the performance of your solution. However, if you believe that your solution is fast enough without using index file, you do not have to generate the file. However, even in such case, your solution should still accept a path to index file as one of the input argument as specified.
8. A record will not be unreasonably long, e.g., you will not see a line that is 10,000+ chars long.
9. Empty records may exist in the original files (before BWT). However, these records will never be matched during searching because the empty string will not be used as a search term when testing your program.

Marking

This assignment is worth 100 points. Below is an indicative marking scheme:

Component	Points
Auto marking	95
Documentation	5

Bonus

Bonus marks (up to 10 points) will be awarded for the solution that achieves 100 points and runs the fastest overall (i.e., the shortest total time to finish **all** the tests). This solution will be shared with the class. Note: regardless of the bonus marks you receive in this assignment, the maximum final mark for the subject is capped at 100.

Submission

Deadline: Sunday 30th April 23:59. Late submission will attract 10% penalty for the first day and 30% penalty for each subsequent day. Use the give command below to submit the assignment:

```
    give cs9319 a2 makefile *.h *.c *.cpp
```

Please use "classrun" to check your submission to make sure that you have submitted all the necessary files.

Plagiarism

The work you submit must be your own work. Submission of work partially or completely derived from any other person or jointly written with any other person is not permitted. The penalties for such an offence may include negative marks, automatic failure of the course and possibly other academic discipline. Assignment submissions will be examined both automatically and manually for such submissions.

Relevant scholarship authorities will be informed if students holding scholarships are involved in an incident of plagiarism or other misconduct.

Do not provide or show your assignment work to any other person - apart from the teaching staff of this subject. If you knowingly provide or show your assignment work to another person for any reason, and work derived from it is submitted you may be penalized, even if the work was submitted without your knowledge or consent. This may apply even if your work is submitted by a third party unknown to you.