**CPP Problem Design**

|  |
| --- |
| **Subject: Word game** |
| **Contributor: 謝公耀, 陳俊儒, 廖宣瑋** |
| **Main testing concept: File I/O**   |  |  | | --- | --- | | **Basics** | **Functions** | | ■ C++ BASICS  □ FLOW OF CONTROL  ■ FUNCTION BASICS  □ PARAMETERS AND OVERLOADING  ■ ARRAYS  □ STRUCTURES AND CLASSES  □ CONSTRUCTORS AND OTHER TOOLS  □ OPERATOR OVERLOADING, FRIENDS,AND REFERENCES  ■ STRINGS  □ POINTERS AND DYNAMIC ARRAYS | □ SEPARATE COMPILATION AND NAMESPACES  ■ STREAMS AND FILE I/O  ■ RECURSION  □ INHERITANCE  □ POLYMORPHISM AND VIRTUAL FUNCTIONS  □ TEMPLATES  □ LINKED DATA STRUCTURES  □ EXCEPTION HANDLING  □ STANDARD TEMPLATE LIBRARY  □ PATTERNS AND UML | |
| **Description:**  A popular word game involves finding words from a grid of randomly generated letters. Words must be **at least three letters long** and formed from adjoining letters. Letters may **not be reused** in one word and it is valid to move across diagonals.  As an example, consider the 4x4 grid of letters below.   |  |  |  |  | | --- | --- | --- | --- | | A | B | C | D | | E | F | G | H | | I | J | K | L | | M | N | O | P |   The word "FAB" is valid (letters in the upper left corner) and the word "KNIFE" is valid. The word "BABE" is not valid because the "B" uses twice in the word. The word "MINE" is not valid because the "E" is not adjacent to the "N".  Write a program that uses a 4x4 two dimensional array to represent the game board. The program should read the words from the text file **words.txt** and then use a recursion algorithm to determine if the word can be formed from the letters on the game board. The program should output all valid words from the file that are on the game board.  **Input:**  Each line contains 4 letters and every 4 lines is a set of value for a game board.  \*\* All inputs are lower case letters.  **Output:**  Find all the valid words on each game board and separate each set by an empty line.  **Sample Input / Output：**   |  |  | | --- | --- | | Sample Input | Sample Output | | abcd  efgh  ijkl  mnop  aabc  defg  hijk  llmn | abc  afb  fie  fin  fink  glop  ink  jim  knife  lop  min  mink  nim  pkg  plonk  pol  polk  abc  abed  afb  baa  baaed  bad  bade  bead  bed  bedim  dab  deaf  deb  dei  die  dill  dim  fad  fade  fed  fide  fie  fill  film  head  heil  hid  hide  hie  hied  hill  him  idea  ill  jill  jim  lid  lie  lied  lief  life  lim  limn  mid  mil  mill | |
| **□ Easy, only basic programming syntax and structure are required.**  **□ Medium, multiple programming grammars and structures are required.**  **■ Hard, need to use multiple program structures or complex data types.** |
| **Expected solving time:**  40 minutes |
| **Other notes:** |