使用 vscode 環境編譯。.vscode 資料夾內含有 launch.json,settings.json,tasks.json.
在 vscode 中直接跑程式。
或在 command line 編譯:
$ g++ -o main.exe main.cpp
$ main.exe
($ ./main.exe 在 vscode 下的 terminal)

輸入 main.exe 後即可使用 search engine，輸入 word 可以得到該 word 所屬的 page(依 page rank 排序)，輸入*end*或 enter 鍵則會結束。

讀檔位置:
input.txt 和 main.cpp 在同個資料夾下。
Pages: ./input/data

預設值: d=0.25, DIFF=0.1;

第一題，我使用兩個二維 vector<vector<int>>分別來儲存一個 page 所指到的對象以及有哪些 pages 指向目前這個 page。因為 page500 是 dead end，所以將他改為指到所有人，除了 page500 之外沒有其他 dead end 或 spider trap。接著使用一維 vector<double>來算 Page rank。算出結果因為要照 page rank 排大小又要記住所屬的 page，所以我另建了一個 class, Page_rank，有兩個 member 存 page 和 page rank。
Time complexity: O(n+m). n: numbers of pages, m:out_links.
Space complexity: O(n+m).

第二題，我使用 map<string, vector<int>>來儲存 word list。每個 word 都對應到一個 vector，紀錄有該 word 的 pages。
Time complexity: O(n*w). n: numbers of pages, w: worlds in a page.
Space complexity: O(n*w).

第三題，使用 list 紀錄 list.txt 裡的字，在 for loop 中，檢查 list 的字是否有出現在上題所算的 word list 中，使用一維 vector<Page_rank>存該字所屬的 pages，並依照 page rank 排序。
Time complexity: O(l*k). l: size of list.txt, k: size of word list.
Space complexity: O(l*k).

Search engine: 使用和第三題一樣的方法查找，不過此時 input 只有一個且不限於在 list.txt 檔案中。

Time complexity: O(k). l: size of list.txt, k: size of word list.
Space complexity: O(k).