CSE 216 – Programming Abstractions (Fall 2019)

Programming Assignment # 4

Submission deadline: Wednesday 20 November 11:59 PM

In this assignment, you will create a spell checker and anagram checker program in both, Python as well as SML languages. Your solution will heavily make use of recursive programming. Kindly find two files worddictionary.txt and speech1.txt in the attachment. Worddictionary.txt contains 466551 words (not necessarily sorted). Speech1.txt is transcript of some speech that contains few words which are mis-spelled and not in worddictionary.txt.

Problem 1 - Spellchecker:

Automated spell-checkers are used to analyze documents and locate words that might be misspelled. These programs work by comparing each word in the document to a large dictionary of words. Any word not found in the dictionary is flagged as potentially incorrect. Write a program to perform spell-checking on a text file. Your program should prompt for a file to analyze and then try to look up every word in the file using binary search. If a word is not found in the dictionary, print it on the screen as potentially incorrect.

Directions to solve problem 1:

- Read all the words from worddictionary.txt and store them in a list.
- Sort the dictionary using RECURSIVE MERGESORT algorithm.
- Write a RECURSIVE binary search algorithm that searches a given word in a dictionary.
- For each word in speech1.txt file use the binary search algorithm to check whether the word is spelled correctly. The case of word doesn't matter.
- Print a list of misspelled words (recursive list printing function required for SML programs).

Rubric:

Feature	**Spellchecker.py**	**Feature**	**Spellcheker.sml**
Reading dictionary and storing in a list	2	Reading dictionary and storing in a list	1
Recursive mergesort algorithm	3	Recursive mergesort algorithm	3
Recursive binary search algorithm	2	Recursive binary search algorithm	2
Print correct list of misspelled words	3	Recursive list printing function	2
		Print correct list of misspelled words	2
Total	***10***	***Total***	***10***

Problem 2 – Anagram checker:

Write a program that solves word anagram problems. An anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once. You will need a large dictionary of English words (see previous problem). Your program generates all anagrams of a particular word and then checks which (if any) are in the dictionary. The anagrams appearing in the dictionary are printed as solutions to the puzzle.

Directions to solve problem 2:

- Write a RECURSIVE program to find all anagrams (permutations) of a given word (upto 8 character long as it takes lot of time to find all permutations of a longer word).
- Read speech1.txt file and create a list of unique words using RECURSIVE duplicate words removal algorithm.
- Read all the words from worddictionary.txt and store them in a list.
- Sort the dictionary using RECURSIVE QUICKSORT algorithm.
- Use the anagrams algorithm to generate all anagrams of each word in unique word list in speech1.txt file.
- Use binary search algorithm to search for each anagram in the dictionary.
- If the anagrams other than the original word appears in the dictionary, print a list of corresponding anagrams.

Rubric:

Feature	**Anagramchecker.py**	**Feature**	**Anagramcheker.sml**
Recursive anagram maker algorithm	3	Recursive anagram maker algorithm	3
Recursive quicksort algorithm	3	Recursive quicksort algorithm	3
Recursive algorithm for duplicate words removal	2	Recursive algorithm for duplicate words removal	2
Print correct list of anagrams found in dictionary	2	Print correct list of anagrams found in dictionary	2
Total	***10***	***Total***	***10***

SML program for reading a text file:

```
val words = String.tokens Char.isSpace o TextIO.inputAll;
val instream = TextIO.openIn "worddictionary.txt";
val wordsList = words instream;
```

Submission:

Submit the following files as separate files. Do NOT submit the worddictionary.txt and speech1.txt files.

a) **Spellchecker.py**
b) **Spellcheker.sml**
c) **Anagramchecker.py**
d) **Anagramcheker.sml**

Evaluation:

The graduate TA will download and run your program and it will be checked for correctness using a few use-cases. If required, the graduate TA will contact you for clarifications.

Note:

Use of iterative functions instead of recursive functions is not permitted. In such case, even the output is correct, entire answer to that problem will get 0 points.

If your code does not compile, it will not be graded.

Late submissions will not be accepted under any circumstances. Submit whatever you can if you were not able to fully complete the assignment.

To be safe, always, ALWAYS, prepare to submit ahead of time, not exactly AT last moment!

Submission deadline: Wednesday 20 November 11:59 PM

Sample output:

Problem 1 - Spellchecker:

```
All incorrect words are:

 ['spich', 'draams', 'comfrot', 'we've', 'lifehacker', 'nataral',
'preditcable', 'don't', 'becume', 'preson', 'it's', 'weren't',
'didn't', 'disracted', 'dobbler', 'aspirtions', 'aksed', 'i'm',
'watned', 'atempt', 'dicipline', 'togther', 'i'm', 'chnaged',
'oponed', 'sometmes', 'miror', 'conseersation', 'can't', 'negtive',
'alwys', 'yuorself', 'wasn't', 'confortable', 'everething', 'comfrot']
```

Problem 2 – Anagram checker:

```
All valid anagrams of  spich are:

 ['chips']

All valid anagrams of  for are:
```

```
 ['ofr', 'orf', 'fro']
All valid anagrams of  succes are:
 ['cusecs']
All valid anagrams of  step are:
 ['sept', 'epts', 'spet', 'pest', 'pets']
All valid anagrams of  out are:
 ['tou']
All valid anagrams of  of are:
 ['fo']
All valid anagrams of  zone are:
 ['ozen', 'zeno']
All valid anagrams of  today are:
 ['doaty', 'toady']
All valid anagrams of  have are:
 ['vahe']
All valid anagrams of  powerful are:
 ['upflower']
All valid anagrams of  speech are:
 ['cheeps']
All valid anagrams of  if are:
 ['fi']
All valid anagrams of  want are:
 ['nawt', 'tawn']
All valid anagrams of  to are:
 ['ot']
All valid anagrams of  goals are:
 ['gaols', 'lagos']
All valid anagrams of  and are:
 ['nad', 'nda', 'adn', 'dan', 'dna']
All valid anagrams of  draams are:
```

```
 ['dramas', 'damars', 'madras']
All valid anagrams of  in are:
 ['ni']
All valid anagrams of  life are:
 ['file', 'fiel', 'feil', 'lief', 'leif']
All valid anagrams of  need are:
 ['ened', 'ende', 'eden', 'dene']
All valid anagrams of  wake are:
 ['weak', 'weka']
All valid anagrams of  up are:
 ['pu']
All valid anagrams of  go are:
 ['og']
All valid anagrams of  beyond are:
 ['boyden']
All valid anagrams of  we are:
 ['ew']
All valid anagrams of  all are:
 ['lal']
All valid anagrams of  comfrot are:
 ['comfort']
All valid anagrams of  is are:
 ['si']
All valid anagrams of  comprise are:
 ['perosmic']
All valid anagrams of  consiste are:
 ['sections']
All valid anagrams of  set are:
 ['est', 'ets', 'ste', 'tse', 'tes']
All valid anagrams of  that are:
```

```
 ['hatt', 'tath']
All valid anagrams of  grown are:
 ['wrong']
All valid anagrams of  accordin are:
 ['draconic', 'cancroid']
All valid anagrams of  alan are:
 ['lana', 'alna', 'anal']
All valid anagrams of  henry are:
 ['rhyne']
All valid anagrams of  the are:
 ['het', 'teh', 'eth']
All valid anagrams of  editor are:
 ['dotier', 'triode', 'rioted']
All valid anagrams of  chief are:
 ['fiche']
All valid anagrams of  nither are:
 ['theirn', 'hinter']
All valid anagrams of  or are:
 ['ro']
All valid anagrams of  bad are:
 ['abd', 'adb', 'bda', 'dba', 'dab']
All valid anagrams of  thing are:
 ['night']
All valid anagrams of  state are:
 ['taste', 'tates', 'testa', 'steat', 'teats']
All valid anagrams of  most are:
 ['stom', 'mots', 'toms', 'mtso']
All valid anagrams of  people are:
 ['popele']
All valid anagrams of  trend are:
```

```
 ['drent']
All valid anagrams of  when are:
 ['hewn']
All valid anagrams of  live are:
 ['ivel', 'vile', 'vlei', 'veil', 'levi', 'evil']
All valid anagrams of  feel are:
 ['fele', 'flee', 'leef']
All valid anagrams of  but are:
 ['btu', 'tub']
All valid anagrams of  with are:
 ['whit']
All valid anagrams of  this are:
 ['hits', 'hist', 'tish', 'isth', 'tshi', 'shit', 'sith']
All valid anagrams of  place are:
 ['clape', 'calpe', 'capel']
All valid anagrams of  ever are:
 ['veer', 'vere', 'reve']
All valid anagrams of  groos are:
 ['sorgo']
All valid anagrams of  here are:
 ['rhee', 'heer']
All valid anagrams of  preditca are:
 ['picrated']
All valid anagrams of  confort are:
 ['crofton']
All valid anagrams of  never are:
 ['enver', 'vener', 'verne', 'nerve']
All valid anagrams of  be are:
 ['eb']
All valid anagrams of  abel are:
```

['bael', 'beal', 'bela', 'elba', 'able', 'bale', 'blae', 'blea', 'albe', 'labe']

All valid anagrams of preson are:

['person', 'speron', 'posner']

All valid anagrams of aspire are:

['spirae', 'spirea', 'sirpea', 'praise', 'paries', 'arispe', 'persia']

All valid anagrams of do are:

['od']

All valid anagrams of not are:

['ont', 'nto', 'ton']

All valid anagrams of time are:

['mite', 'meit', 'item', 'emit']

All valid anagrams of words are:

['sword']

All valid anagrams of one are:

['noe', 'neo', 'eon', 'eno']

All valid anagrams of every are:

['veery', 'verey']

All valid anagrams of day are:

['ady', 'yad']

All valid anagrams of scares are:

['carses', 'caress', 'crases', 'escars', 'seracs']

All valid anagrams of what are:

['wath', 'thaw']

All valid anagrams of afraid are:

['farida']

All valid anagrams of at are:

['ta']

All valid anagrams of point are:

['pinot', 'pinto', 'piton']

```
All valid anagrams of  my are:
 ['ym']
All valid anagrams of  was are:
 ['asw', 'swa', 'saw']
All valid anagrams of  stuck are:
 ['tucks']
All valid anagrams of  desired are:
 ['resided', 'derides']
All valid anagrams of  take are:
 ['kate', 'keta', 'keat', 'teak', 'etka']
All valid anagrams of  action are:
 ['cation', 'atonic', 'tonica', 'actino']
All valid anagrams of  disracte are:
 ['acridest']
All valid anagrams of  some are:
 ['osme', 'mose', 'meso']
All valid anagrams of  write are:
 ['twire', 'twier']
All valid anagrams of  self are:
 ['fels']
All valid anagrams of  abut are:
 ['buat', 'tabu', 'taub', 'tuba']
All valid anagrams of  dreams are:
 ['dermas', 'madres']
All valid anagrams of  aksed are:
 ['asked']
All valid anagrams of  stefan are:
 ['fasten', 'nefast']
All valid anagrams of  are are:
 ['rae', 'rea', 'aer', 'ear', 'era']
```

```
All valid anagrams of  hopes are:
 ['phose', 'shope']
All valid anagrams of  as are:
 ['sa']
All valid anagrams of  much are:
 ['chum']
All valid anagrams of  watned are:
 ['wanted']
All valid anagrams of  me are:
 ['em']
All valid anagrams of  left are:
 ['flet', 'felt']
All valid anagrams of  no are:
 ['on']
All valid anagrams of  choice are:
 ['echoic']
All valid anagrams of  simply are:
 ['limpsy']
All valid anagrams of  must are:
 ['smut', 'stum', 'muts', 'tums']
All valid anagrams of  on are:
 ['no']
All valid anagrams of  please are:
 ['elapse', 'asleep', 'sapele']
All valid anagrams of  way are:
 ['yaw']
All valid anagrams of  years are:
 ['ayers', 'ayres', 'eyras', 'reasy', 'resay', 'seary', 'sayer',
'sayre']
All valid anagrams of  lack are:
 ['calk', 'kcal']
```

```
All valid anagrams of  tells are:
 ['stell']
All valid anagrams of  everythi are:
 ['thievery']
All valid anagrams of  know are:
 ['kwon', 'wonk']
All valid anagrams of  really are:
 ['rallye', 'ellary']
All valid anagrams of  found are:
 ['fondu']
All valid anagrams of  us are:
 ['su']
All valid anagrams of  so are:
 ['os']
All valid anagrams of  tired are:
 ['tried', 'diter']
All valid anagrams of  seeing are:
 ['signee', 'genies']
All valid anagrams of  soar are:
 ['osar', 'oars', 'asor', 'sora', 'orsa', 'oras', 'rosa']
All valid anagrams of  just are:
 ['juts']
All valid anagrams of  see are:
 ['ese']
All valid anagrams of  them are:
 ['hemt', 'meth']
All valid anagrams of  dashed are:
 ['shaded']
All valid anagrams of  chnaged are:
 ['changed', 'ganched']
```

```
All valid anagrams of  it are:
 ['ti']
All valid anagrams of  oponed are:
 ['podeon']
All valid anagrams of  doing are:
 ['dingo', 'godin', 'gondi', 'gonid']
All valid anagrams of  necessar are:
 ['casernes']
All valid anagrams of  look are:
 ['kolo']
All valid anagrams of  an are:
 ['na']
All valid anagrams of  honest are:
 ['stheno', 'ethnos']
All valid anagrams of  conseers are:
 ['necroses']
All valid anagrams of  lot are:
 ['otl', 'tlo', 'tol']
All valid anagrams of  stories are:
 ['sorties', 'sorites', 'rosiest', 'trioses', 'isoster', 'rossite']
All valid anagrams of  why are:
 ['hwy']
All valid anagrams of  they are:
 ['hyte', 'hyet', 'yeth']
All valid anagrams of  these are:
 ['sheet']
All valid anagrams of  keep are:
 ['peke', 'peek']
All valid anagrams of  despair are:
 ['aspired', 'diasper', 'diapers', 'pardesi', 'presaid', 'praised']
```

```
All valid anagrams of  once are:
 ['cone', 'coen', 'econ']
All valid anagrams of  said are:
 ['aids', 'sida', 'idas', 'sadi', 'adis', 'dasi', 'dais', 'disa',
'dias']
All valid anagrams of  naked are:
 ['knead', 'kande', 'kaden', 'danke']
All valid anagrams of  truth are:
 ['thurt']
All valid anagrams of  alwys are:
 ['yawls']
All valid anagrams of  than are:
 ['hant', 'tanh', 'anth', 'nath']
All valid anagrams of  lie are:
 ['ile', 'lei', 'eli']
All valid anagrams of  get are:
 ['gte', 'teg']
All valid anagrams of  real are:
 ['eral', 'earl', 'arel', 'rale', 'arle', 'lear', 'lare']
All valid anagrams of  yuorself are:
 ['yourself']
All valid anagrams of  free are:
 ['reef', 'fere', 'feer']
All valid anagrams of  more are:
 ['orme', 'orem', 'rome', 'omer', 'mero']
All valid anagrams of  put are:
 ['utp', 'tup']
All valid anagrams of  there are:
 ['three', 'rethe', 'theer', 'heter', 'ether']
All valid anagrams of  fears are:
 ['fares', 'farse', 'frase', 'safer']
```

```
All valid anagrams of  am are:
 ['ma']
All valid anagrams of  pushing are:
 ['shuping', 'gunship']
All valid anagrams of  limits are:
 ['mislit']
All valid anagrams of  actions are:
 ['cations', 'atonics', 'costain']
All valid anagrams of  eddie are:
 ['dedie']
All valid anagrams of  harris are:
 ['arrish', 'rarish', 'shirra', 'sirrah']
All valid anagrams of  jr are:
 ['rj']
All valid anagrams of  sooner are:
 ['nooser', 'seroon']
All valid anagrams of  from are:
 ['form']
All valid anagrams of  other are:
 ['toher', 'thoer', 'theor', 'thero', 'thore', 'throe']
All valid anagrams of  side are:
 ['ides', 'dies', 'desi', 'deis', 'ised', 'seid', 'esdi']
All valid anagrams of  fear are:
 ['afer', 'fare', 'fera', 'frae', 'rafe']
All valid anagrams of  ready are:
 ['deray', 'deary', 'rayed', 'yeard']
```