

CSE 1301 PRACTICAL SESSION 8

Strings and File I/O (16 marks)

Aim: By the end of this practical session, you should be able to write programs which use string manipulation functions, have functions with string parameters, and perform input/output to and from files.

Coding style and documentation: You are expected to document your programs and to use a standard, clear, and consistent coding style. Up to 2 of the marks for each practical class may be deducted for poor coding style and/or inadequate documentation.

Preparation (to be completed before class) **(2 marks)**

Write all the algorithms required for the questions below, and attempt the code for all the questions.

PART 1: Which is longer? **(5 marks)**

- (a) Write a function, `strlen()` that returns the number of characters in a string, up to, but not including the first null character. Your function is not to use `string.h` library functions. Apply this function to the remaining questions of Part 1.
- (b) Write a function, `compareLen()` that takes two string parameters and returns a positive number if the first string is longer than the second, 0 if they are the same length, and a negative value if the second string is longer.
- (c) Write a function, `iAmLonger()` that takes two string parameters and returns a string which is a pointer to the longer of the string parameters. It should call `compareLen()`.
- (d) Write a main program with a query loop that allows the user to test the two above functions. Construct a test data set that tests all paths through your code.

PART 2: Substitution **(4 marks)**

- (a) Write a function, `substituteChar()` that takes three parameters - two characters and a string. This function should search the string for copies of the first character, and change them all to the second character. The function should return the number of times a substitution was made.
- (b) Write a main program to allow the user to replace multiple letters in the same sentence, using a loop. For each iteration, your program should prompt the user to enter two characters, the letter to find and its replacement. Construct a test data set that tests all paths through your code, including the main program and the function.

PART 3: Formatting**(5 marks)**

Write a program that reads in the name of an input file, the name of an output file and a positive integer N. The program should then open the input file for reading, and the output file for writing. Your algorithm should read in the words contained in the input file and reformat them into left-justified lines with a maximum of N characters per line, which will be stored in the output file. For example, given the input file (available from <http://www.csse.monash.edu.au/courseware/cse1301/pracs/prac08-3.txt>):

```
Far back in the mists of ancient time,  
in the great and glorious days of the former galactic empire,  
    life was wild, rich and  
(on the whole)  
tax-free.
```

```
Spirits were brave, the stakes were high,  
  
men were Real Men, women were Real Women,  
and small furry creatures from Alpha Centauri were  
Real Small Furry Creatures From Alpha Centauri.
```

and an N value of 60, your program should produce an output file containing:

```
Far back in the mists of ancient time, in the great and  
glorious days of the former galactic empire, life was wild,  
rich and (on the whole) tax-free. Spirits were brave, the  
stakes were high, men were Real Men, women were Real Women,  
and small furry creatures from Alpha Centauri were Real  
Small Furry Creatures From Alpha Centauri.
```

Notes:

- Your program must fill as much of each line as possible, while avoiding breaking a word over two lines. However, if a word is too long to fit on a single line (i.e. it is longer than N), then it must be broken over two or more lines.
- Words ending in a period ('.') are to be followed by two space characters, while all other words are separated by a single space. You should avoid including space characters at the end of a line. However you will not be penalised if you do so, even if the space characters cause the line to be longer than N.
- There are two main methods you can use: deal with (1) a single character at a time, or (2) a single word at a time. You may find it easier to work with a single word at a time.
- Words in the input file are separated by at least one *white space*: a space, a tab ('\t'), a carriage return ('\r'), or a new line ('\n') character, or any combination of these.
- Your program should check that the value of N is valid, and check for input/output errors during file operations (e.g., open error, read error, etc).
- Remember that software development should be an incremental process. Start by solving the more simple problems (e.g. for N larger than the longest word), and once solved, work towards the more difficult (e.g. splitting long words over multiple lines).

Submission:

Preparation: Show written preparation work to your demonstrator at start of class for marking. If not done before class, it will not receive any marks, however you should still show it to your demonstrator during class to check your understanding. Show your pracs to your demonstrator for marking.

PART 1, 2, 3: To be marked by the demonstrator during this class. No late submission.

Additional Component**(2 BONUS MARKS)**

- (a) Write a program that uses random number generation to create sentences. The program should use four arrays of pointers to char called `article`, `noun`, `verb` and `preposition`. The program should create a sentence by selecting a word at random from each array in the following order: `article`, `noun`, `verb`, `preposition`, `article` and `noun`. As each word is picked, it should be concatenated to the previous words in an array large enough to hold the entire sentence. The words should be separated by spaces. When the final sentence is output, it should start with a capital letter and end with a period. The program should generate 20 such sentences and output them to a file.

The arrays should be filled as follows:

The `article` array should contain the articles "the", "a", "one", "some" and "any"; the `noun` array should contain the nouns "boy", "girl", "dog", "town" and "car"; the `verb` array should contain the verbs "drove", "jumped", "ran", "walked" and "skipped"; the `preposition` array should contain the prepositions "to", "from", "over", "under" and "on".

- (b) Modify your solution to question (a) by adding at least three more words to the overall vocabulary.