THE UNIVERSITY OF WESTERN AUSTRALIA

DESK No. | | | |

FAMILY NAME: _____

GIVEN NAMES: _____

SIGNATURE: _____

STUDENT NUMBER: | | | | | | | | |

SEMESTER 2, 2018 SUPPLEMENTARY AND DEFERRED EXAMINATIONS

CITS2002

Physics, Mathematics & Computing

ECM

Systems Programming

Programming and Systems

This paper contains: **6** Pages **(including title page)**

Time Allowed: **2:00** hours

INSTRUCTIONS:
Answer all 6 questions. Each question is work 10 marks. You should attempt all questions.

THIS IS A CLOSED BOOK EXAMINATION

SUPPLIED STATIONERY	ALLOWABLE ITEMS
1 x Answer Booklet 18 Pages	**UWA Approved Calculator with Sticker**

PLEASE NOTE

Examination candidates may only bring authorised materials into the examination room. If a supervisor finds, during the examination, that you have unauthorised material, in whatever form, in the vicinity of your desk or on your person, whether in the examination room or the toilets or en route to/from the toilets, the matter will be reported to the head of school and disciplinary action will normally be taken against you. This action may result in your being deprived of any credit for this examination or even, in some cases, for the whole unit. This will apply regardless of whether the material has been used at the time it is found. Therefore, any candidate who has brought any unauthorised material whatsoever into the examination room should declare it to the supervisor immediately. Candidates who are uncertain whether any material is authorised should ask the supervisor for clarification.

Candidates must comply with the Examination Rules of the University and with the directions of supervisors.
No electronic devices are permitted during the examination.
All question papers and answer booklets are the property of the University and remain so at all times.

This page has been left intentionally blank

1) Consider the C99 function `isSubset()`, whose prototype follows:

```
bool isSubset(char set1[], int len1, char set2[], int len2);
```

The function receives two arrays of characters, named `set1` and `set2`, whose lengths are provided by parameters `len1` and `len2`, respectively.

All elements of `set1` are guaranteed to be unique, but are not necessarily sorted. The same is also true of the elements of `set2`.

There is no requirement that `len1` and `len2` have the same value.

The goal of the function is to determine if `set2` is a non-empty subset of `set1`.

Write the function `isSubset()` in C99.

(10)

2a) Different applications for processing large text documents (such as a book's chapter) may sometimes prefer a chapter's text to be available as a single (long) string and, at other times, prefer the chapter to be available as a sequence of sentences. This question requires you to write two functions to perform the conversion between chapters and sentences.

Write a function in C99, named `makeSentences()`, that converts a chapter into a vector of sentences. The function should match the following prototype:

```
char **makeSentences(char *chapter, int *nSentences);
```

The function receives a single string (`chapter`) that points to the chapter's text. Assume that all sentences end with a full-stop character ('.'), and that `chapter` is guaranteed to end with a complete sentence.

The function produces 2 values. The function returns all sentences as a dynamically allocated array of dynamically allocated strings. The number of sentences is returned via the function's second parameter (`nSentences`). If any call to dynamically allocate memory fails, the function should return NULL.

(5)

2b) Write a function in C99, named `makeChapter()`, that converts a vector of sentences into a single string that represents the chapter. The function should match the following prototype:

```
char *makeChapter(char **sentences, int nSentences);
```

The function receives a vector of sentences (as might be produced by `makeSentences()`, above) and converts the sentences into a single dynamically allocated string holding a whole chapter of sentences. When (re)constructing the chapter, ensure that you add a single space after each complete sentence. If any calls to dynamically allocate memory fail, `makeChapter()` should return NULL.

You should assume that `nSentences` is always strictly greater than zero, and that `sentences` contains `nSentences` elements. Assume also that each string in `sentences` represents a complete sentence.

(5)

3a) Explain clearly the following state transitions for processes and the reasons for the transitions:

1. from Running to Blocked.

2. from Blocked to Blocked-Suspend.

(5)

3b) Explain clearly the difference between a *program* and a *process*. What are the important pieces of information that a multi-processing operating system needs to save during process switching?

(5)

4a) Explain the importance of logical to physical address translation.

Consider a 20-bit computer in which the logical address has a 12-bit offset. Explain clearly, with the aid of a diagram, how the logical to physical address translation is performed for this computer.

What is the maximum number of pages that a process can be allocated in this computer?

(5)

4b) With the aid of diagrams, explain the differences between *internal* memory fragmentation and *external* memory fragmentation.

(5)

5a) With reference to a diagram, explain the actions of a Unix-based operating system when a process invokes the *fork()* system-call.

(5)

5b) Explain how an operating system's use of virtual memory can enable the operating system to appear to support the use of more memory than is physically installed in a computer.

(5)

---

6) Consider the function `countEntries()`, whose prototype follows:

```
void countEntries(char *dirName, int *nFiles, int *nDirs);
```

The parameter named `dirName` points to a character string, providing the name of a directory to be recursively explored for files and subdirectories.

You may assume that `dirName`, and its subdirectories, contain only files and directories.

After `countEntries()` has explored everything below the named directory, the parameters `nFiles` and `nDirs` should provide, to the calling function, the number of files and directories found, respectively.

If `countEntries()` finds a directory that cannot be opened, it should report an error and continue its processing (i.e. the function should not exit on such an error).

Write the function `countEntries()` in C99.

(10)

END OF PAPER