THE UNIVERSITY OF WESTERN AUSTRALIA

Computer Science and Software Engineering

SEMESTER 2, 2017 DEFERRED EXAMINATIONS

CITS2002
Systems Programming

FAMILY NAME:_____ GIVEN NAMES: _____

STUDENT ID: | | | | | | | | | SIGNATURE: _____

This Paper Contains: **5** pages **(including title page)**
Time allowed: **2:00 hours (including reading time)**

INSTRUCTIONS:

Answer all 6 questions. Each question is worth 10 marks.

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.

Supervisors Only - Student left at:

THIS PAGE INTENTIONALLY LEFT BLANK

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

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

The function receives two arrays of integers, 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)

---

2) Consider a 2-dimensional spreadsheet – a rectangular grid of cells, each of which holds an arbitrary string of characters, or an empty string if the value of the cell has not yet been defined. Each cell is identified by 2 non-negative integer values, identifying a specific row and column.

Define a C99 data structure to define an instance of the 2-dimensional spreadsheet.

Next, define and implement three C99 functions:

- one to return a pointer to allocated memory suitable for storing and managing the spreadsheet,
- one to swap the contents of two indicated columns in a spreadsheet, and
- one to completely deallocate all memory allocated to a spreadsheet.

(10)

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) With reference to two distinct examples, explain *The Principle of Referential Locality*, and why it is important to operating-system design.

(5)

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

Consider a 16-bit computer in which the logical address has a 10-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) What are *system calls*, and why are they important in the design and implementation of an operating system?

Explain why almost every process needs to execute at least one system call during its execution.

(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 clearly how virtual memory helps in improving processor utilization in a multiprogramming system.

(5)

---

6) Consider the following operating system dependent system-calls:

```
int unlink(char *filename);
int  rmdir(char *directoryname);
```

The `unlink()` system-call may be used to remove a file from its parent directory, and the `rmdir()` system-call may be used to remove a directory from its parent directory provided that `directoryname` itself contains no other files or directories.

Assume that both system-calls return 0 on success, and `-1` on failure.

Using the `unlink()` and `rmdir()` system-calls, write a C99 function named `removeDirectory()` that removes the indicated (potentially non-empty) directory and all of its contents. Your function should have the following prototype:

```
int removeDirectory(char *directoryname);
```

You should assume `directoryname` contains only files and directories.

Your function should attempt to remove as many files and sub-directories as possible, returning 0 on complete success and non-zero otherwise. If `directoryname` could not be opened as a directory, your function should return `-1`.

(10)

END OF PAPER