CS2014-2

Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Faculty of Engineering, Mathematics and Science

School of Computer Science and Statistics

Integrated Computer Science **Trinity Term 2016**
Computer Science and Business
Senior Freshman Annual Examinations

Systems Programming

Wednesday 18th May 2016 **Sports Centre** **14.00 – 16.00**

Dr David Gregg

Instructions to Candidates:

- Answer 2 out of the 3 questions
- All questions are marked out of 50
- All program code should be commented, indented and use good programming style

Materials permitted for this examination:

- Calculator

1. Write a C function that identifies whether or not a string is a valid email address. For the purposes of this question (the real rules are more much complex) an email address may be defined as follows. An email address consists of a sequence of:
   a. one or more identifiers separated by full stops, followed by
   b. the @ symbol, followed by
   c. one or more identifiers separated by full stops, followed by
   d. a full stop, followed by
   e. a terminator, followed by
   f. the end of the string

   The function should have the following prototype:

   int isValidEmailAddress(char email_address[ ], char * terminators[ ], int nterms);

   An identifier is a sequence of one or more alphanumeric characters. You may use the functions isalpha(char) and isdigit(char) to identify alphabetic and numeric characters respectively. A terminator is a final string, such as "com", "net" or "ie" that ends an email address. The set of all valid terminators is stored in the array of strings terminators. There are *nterms* strings in the terminators array. It is guaranteed that all terminators are also valid identifiers; that is terminators will never include characters that are not alphanumeric.

   [50 marks]

2. You are asked to write an application that needs to keep track of sets of web addresses (URLs). Write a C abstract data type (ADT) to represent sets of strings, where the hash function is suitable for hashing URLs. Your ADT should provide functions to:

    - create a new empty set
    - add a string to the set
    - remove a string from the set
    - check whether a given string is a member of the set
    - compute the union of two sets of strings
    - compute the intersection of two strings
    - free the memory used by the set

[50 marks]

3. On most computers the smallest sized piece of memory that can be accessed is a byte. However, for some applications it is useful to be able to operate on nybbles, where a nybble is four bits, or half a byte. When we store arrays of nybbles, ideally we would be able to pack two nybbles into each byte in memory.

   Write a C abstract data type (ADT) to represent a sequence of nybbles that can be indexed like an array. Your ADT should store the nybbles internally by packing two nybbles into each byte of storage. Your ADT should support the following functions:

```
// create new array of nybbles
struct nybble_array * nybble_array_new(int size);
// return the nybble value at position index
int get_nybble_value(struct nybble_array * this, int index);
// set the nibble at position index to value
void set_nybble_value(struct nybble_rray * this, int index, int value);
// free the memory used by a nibble array
void nybble_array_free(struct nibble_array * this);
```

[50 marks]