NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 1 AY2011/2012

CS1010 – PROGRAMMING METHODOLOGY

November 2011 Time allowed: 2 hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper consists of **SIX (6)** questions and comprises **TEN (10)** printed pages.
2. This is an **OPEN BOOK** examination.
3. Answer all questions.
4. Write your answers in the ANSWER SHEETS provided.
5. Fill in your Matriculation Number with a pen, clearly on every page of your ANSWER SHEETS.
6. You may use **2B pencil** to write your codes. Pen is preferred for other questions.
7. Note the penalty will be given for codes that are unclear or unnecessarily long.
8. You must submit only the ANSWER SHEETS and no other document.

Q1. Multiple Choice Questions (MCQs) [12 marks]

Each MCQ has one correct answer and is worth 2 marks. There is no penalty for wrong answer.

Q1.1 Which of the following statements is/are true about working in sunfire?

i. To disconnect from sunfire, we can issue command **`logout`** or **`exit`**.
ii. To compile a C program using **`gcc`**, we must include the option **`-Wall`**.
iii. The **`rename`** command is used to rename a file.

A. (i) only
B. (ii) only
C. (iii) only
D. (i) and (ii) only
E. (i) and (iii) only

Q1.2 What is printed out by the following C code fragment?

```
char c = 'b', d = 'H';
int a = d - toupper(c);
printf("%c %d\n", c, a);
```

A. `B -16`
B. `B 6`
C. `b -16`
D. `b 6`
E. It will give compile-time error.

Q1.3 What is the largest value that could be returned by the following function?

```
int func(int key)
{
    int index, arr[] = {2, 4, 6, 8, 10, 12, 14};
    for (index = 0; index < 7; index++)
        if (arr[index] > key)
            break;
    return index;
}
```

A. 2
B. 3
C. 7
D. 8
E. None of the above.

Q1.4 Suppose a text file "numbers.txt" contains the following values:

```
2 3
7
-5 2 8
```

What is printed out by the following C code fragment?

```
FILE *infile;
int i, j, foo = 0;

infile = fopen("numbers.txt", "r");
while (fscanf(infile, "%d %d", &i, &j) > 0)
    foo = (i > j) ? i : j;

printf("%d\n", foo);
```

A. 0

B. 7

C. 8

D. It will give compile-time error.

E. It will give run-time error.

Q1.5 What does the following function return?

```
int mystery(int x, int y)
{
   if (x == 0)
      return y;
   else if (x < 0)
      return mystery(++x, --y);
   else
      return mystery(--x, ++y);
}
```

A. It returns the value of y.

B. It returns the value of x – y.

C. It returns the value of x + y.

D. It returns the value of x * y.

E. It will give compile-time error.

Q1.6 What is printed out by the following code fragment?

```
double num = (double) rand() / RAND_MAX * 6 + 1;
printf("%d\n", (int) num * 2 + 1);
```

A. It always prints 3

B. It prints either 3 or 13

C. It prints either 3 or 15

D. It prints a random odd integer in the range [3, 13] (both inclusive)

E. It prints a random odd integer in the range [3, 15] (both inclusive)

Q2. For each of the following programs, write out its output. **[15 marks]**

a. [4 marks]

```
#include <stdio.h>
#define N 9

void func(int, int *, int [], int);

int main(void)
{
    int a = 0, b[N] = {3};

    func(a, &b[1], b, b[2]);
    printf("%d %d %d %d\n", a, b[0], b[1], b[2]);

    return 0;
}

void func(int a, int *b, int c[], int d)
{
    a = 1;
    (*b)++;
    c[1] += 5;
    d--;
}
```

Q2. (continued…)

b. [5 marks]

```
#include <stdio.h>
#include <string.h>
#define N 5

int main(void)
{
    int i, j;
    char *temp;
    char *fruits[N] = {"apple", "mango", "pineapple",
                       "orange", "banana"};

    for (i=1; i<N; i++)
    {
        for (j=0; j<N-i; j++)
        {
            if ( strcmp(fruits[j], fruits[j+1]) < 0 )
            {
                temp = fruits[j];
                fruits[j] = fruits[j+1];
                fruits[j+1] = temp;
            }
        }
    }

    for (i=0; i<N; i++)
        printf("%s\n", fruits[i]);

    return 0;
}
```

Q2. (continued…)

c. [6 marks]

```
#include <stdio.h>
#include <string.h>

typedef struct
{
    char name[10];
    int age;
} person;

void func1(person *, char [], int);
void func2(person []);
void func3(person);

int main(void)
{
    person data[] = {{"Zhou", 25}, {"Tamil", 22},
                     {"Potter", 33} };
    func1(&data[0], "Ismail", 15);
    printf("%s %d\n", data[0].name, data[0].age);
    func2(data);
    printf("%s %d\n", data[1].name, data[1].age);
    func3(data[2]);
    printf("%s %d\n", data[2].name, data[2].age);
    return 0;
}

void func1(person *ptr, char name[10], int age)
{
    strcpy(ptr->name, name);
    ptr->age = age;
}

void func2(person per[])
{
    int i;
    for (i=0; i<3; i++)
        per[i].age++;
}

void func3(person per)
{
    strcpy(per.name, "Ace");
    per.age--;
}
```

Q3. In many word games, a letter in a word is scored according to its point, which is inversely proportional to its frequency in English words. In a certain word game, the points are allocated as follows:

Points	Letters
1	A, E, I, L, N, O, R, S, T, U
2	D, G
3	B, C, M, P
4	F, H, V, W, Y
5	K
8	J, X
10	Q, Z

Write a function called **`compute_score(char *word)`** that takes in a word which is a string comprising only uppercase letters, and returns its score, which is the total points of all the letters in the word.

For example, the word "EXAM" is worth 13 points: 1 for E, 8 for X, 1 for A, and 3 for M.

[7 marks]

Q4. Write a recursive function **`get_min_index`** that will return the index of the smallest element of an array *arr* of *n* integers, where *n* > 0. If there are several occurrences of the smallest element, then the index of the first occurrence is returned.

For example, for this array defined in the main function:

```
int arr[10] = {20, 10, 4, 22, 6, 30, 4, 35, 99, 55};
```

the function returns the value of 2, since arr[2] is the first occurrence of the smallest element in the array.

Note that the above array is just an example. Your function should be able to work on an integer array containing any values in general.

Fill in the incomplete statement in the main function on the Answer Sheet, and write the function **`get_min_index`**. [8 marks]

Q5. [Total: 20 marks]

A C program reads student data from a file, then answers some queries. The following constants have been defined:

```
#define NAME_LENGTH 30
#define MAX_NUM_COURSES 20
#define MAX_NUM_STUDENTS 100
```

A structure has also been defined as shown below and you must not change it.

```
typedef struct {
   char name[NAME_LENGTH];
   int numCourses;
   int scores[MAX_NUM_COURSES];
} student_t;
```

The above structure definition consists of three components:

i. A name. The name is an array of characters of 30 elements. You may assume that the name does not contain any space character.

ii. Number of courses. This is an integer. A student can take at least one and at most 20 courses.

iii. Score in each course. Each score is an integer. Since a student can take at most 20 courses, the scores are stored in an array of 20 elements.

a. Read student data from a file. Assume that the class has at most 100 students but there may be fewer students. Write a function called

```
int readStudents(student_t students[], char *fileName)
```

The function opens the file whose name is stored in fileName. This is a text file that contains the records of the students in the class.

After opening the file, the function reads a student's name, the number of courses the student takes, and the scores of the courses. If there are more students in the file, the function keeps reading until it reaches the end of the file. It then closes the file and returns the number of students read. [7 marks]

b. Write a function that determines whether a student is in the record:

```
int findStudentByName(student_t students[],
                      int numberStudents, char name[])
```

If among the students, one (or more) has the name given as the third argument, the function returns the index of the first student whose name matches. The index should be between zero and numberStudents – 1 (inclusively). If no student has this name, the function returns -1. [7 marks]

c. Write a function that returns the average score of a student:

```
float averageScore(student_t stu)
```

This function calculates and returns the average score of this student. [6 marks]

Q6. [Total: 18 marks]

A local entrepreneur wishes to develop a new social network system, called *iLink*, and she employs you to help develop programs to handle friendship relation service. In modeling the friendship relation, you have adopted a two-dimensional array representation, in which the array is of size MAXSIZE × MAXSIZE. This array is called **friendArr**. You have also decided that **friendArr** will NOT be a global variable.

A simplified version of **friendArr** with 6 users is given below:

	0	1	2	3	4	5
0	1	0	1	0	0	0
1	0	1	0	0	0	1
2	1	0	1	1	0	0
3	0	0	1	1	0	0
4	0	0	0	0	1	1
5	0	1	0	0	1	1

Under this representation, you set the entry (*i*, *j*) of **friendArr** to 1 if the user identified by *i* has added the user identified by *j* as a **direct friend**. Otherwise, entry (*i*, *j*) should contain 0. By default, an *iLink* user will always add himself/herself as a direct friend, and the **friendArr** has the following symmetry property:

Value at entry (*i, j*) = *Value at entry* (*j, i*)

For the questions below, you are allowed to introduce auxiliary function(s) to help define the functions required.

a. Write a function **`iSolitude`** that displays a list of users (represented by the respective array indices) who have added the **LEAST** number of direct friends. For instance, for the small **friendArr** shown above, **`iSolitude`** will print out users 0, 1, 3 and 4 (not necessarily in that order), as they have the smallest number of direct friends (each of them has only two direct friends, including himself/herself).

[6 marks]

b. The entrepreneur has also requested that you compute the **friend-of-friend** relation, so that if *u* and *v* are direct friends of each other, *iLink* can introduce other direct friends of *u* to *v*, and vice versa. Specifically, *i* and *j* have a **friend-of-friend** relationship if and only if the following two conditions hold:

   i. *j* is NOT a **direct friend** of *i*; and
   ii. there exists a distinct user *k* who is a **direct friend** of both *i* and *j*.

   Write a function **`uFriend`** that displays all pairs (*i*, *j*) of **friendArr** such that user *i* is a friend-of-friend of user *j*. In the small **friendArr** array shown above, (0, 3) has friend-of-friend relationship, as 0 and 3 are not direct friend of each other, and user 2 is a direct friend of both 0 and 3. [6 marks]

Q6. (continued…)

c. This morning, the entrepreneur passed you the following algorithm, and requested that you use it to create from **friendArr** a new array, which is of the same size as **friendArr**.

```
Begin
    for each i from 0 to maxsize – 1
        newArray[ i ][ i ] ← 1
    for each i from 0 to maxsize – 1
        for each j from 0 to maxsize – 1
            if i ≠ j then newArray[ i ][ j ] ← friendArr[ i ][ j ]
    for each k from 0 to maxsize – 1
        for each i from 0 to maxsize – 1
            for each j from 0 to maxsize – 1
                newArray[ i ][ j ] ← newArray[ i ][ j ] or
                                     ( newArray[ i ][ k ] and newArray[ k ][ j ] )
End
```

i. You trace the algorithm on paper by using the small **friendArr** given above, which is replicated here:

	0	1	2	3	4	5
0	1	0	1	0	0	0
1	0	1	0	0	0	1
2	1	0	1	1	0	0
3	0	0	1	1	0	0
4	0	0	0	0	1	1
5	0	1	0	0	1	1

Show the final content of **newArray** on the Answer Sheet. [3 marks]

ii. Describe what the algorithm is supposed to do for the *iLink* system. [3 marks]

=== END OF PAPER ===