NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 1 AY2010/2011

CS1010E — PROGRAMMING METHODOLOGY

Nov / Dec 2010 Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper consists of **FOURTEEN (14)** questions and comprises **EIGHTEEN (18)** printed pages, including this page.
2. Answer **ALL** questions.
3. Answer Section A (Questions 1 to 10) by shading the letter corresponding to the most appropriate answer on the OCR form provided.
4. Answer Section B (Questions 11 to 14) within the space provided in this booklet. You may use pen or pencil to write your answers.
5. This is an **OPEN BOOK** exam. The maximum mark is **40**.
6. Calculators are allowed, but not electronic dictionaries, laptops, PDAs, or other computing devices.
7. Do not look at the questions until you are told to do so.
8. Please write your **Matriculation number** below.

This portion is for examiner's use only.

Question	Marks	Remarks
Q11		
Q12		
Q13		
Q14		
Total		

SECTION B (4 Questions : 30 Marks)

Write your answers in the space provided.

11. [**6 marks**] Given an **unsorted** integer array of unique values and an integer z, we would like to determine if there are two **distinct** elements in the array with values x and y so as to satisfy $x \times y = z$.

    Write the function `findXandY` that takes in an array `A` of `n` elements and the value `z`. The function returns `TRUE` if x and y can be found; or `FALSE` otherwise. If x and y can be found, these two values are returned via the output parameters `x` and `y`. Assume the definition of the following symbolic constants.

    ```
    #define TRUE  1
    #define FALSE 0
    ```

    Note: If you require the use of any sorting and/or searching functions covered during the lectures, you may call the function directly without implementing it.

    Answer:

    ```
    int findXandY(int A[], int n, int z, int *x, int *y)
    {
    ```

12. Study the following function.

```
void chopChop(char str[], char special)
{
   int length, start = 0, end = 0;

   length = strlen(str);
   while (end < length)
   {
      if (str[end] == special)
      {
         str[end] = '\0';
         printf("%s\n", &str[start]);
         start = end+1;
      }
      end++;
   }
   if (start < length)
      printf("%s\n", &str[start]);
}
```

(a) [**3 marks**] What is the output of the following function call? Fill in each box with a single output character including blank spaces or blank lines.

```
char str[40] = "abc AA fA g";
chopChop(str,' ');
```

Answer:

Line #1												
Line #2												
Line #3												
Line #4												
Line #5												

(b) [**3 marks**] What is the output of the following function call? Fill in each box with a single output character including blank spaces or blank lines.

```
char str[40] = "abc AA fA g";
chopChop(str,'A');
```

Answer:

Line #1												
Line #2												
Line #3												
Line #4												
Line #5												

13. (a) **[2 marks]** Given an amount in cents, we would like to determine the **least** number of coins in 50¢, 20¢ and 10¢ denominations. Assume that the amount can always be completely represented using these three denominations.

Write the function `getCoins` that takes in `amount` and returns the number of coins of each denomination via the output parameters `cent50`, `cent20` and `cent10`.

Answer:

```
void getCoins(int amount, int *cent50, int *cent20, int *cent10)
{
```

(b) **[4 marks]** Given an integer array of n values representing amounts in cents, sort the array in order of increasing number of 50¢ coins. In cases where there are the same number of 50¢ coins, order them in increasing number of 20¢ coins. Similarly, in cases where there are the same numbers of 50¢ cents and 20¢ coins, order them in increasing number of 10¢ coins. The following `sortAmount` function is given.

```
void sortAmount(int amounts[], int n)
{
   int i, j, minIndex, temp;

   for (i = 0; i < n-1; i++)
   {
      minIndex = i;
      for (j = i+1; j < n; j++)
         // Note the usage of the compareAmount function
         if (compareAmount(amounts[j], amounts[minIndex]) < 0)
            minIndex = j;

      temp = amounts[i];
      amounts[i] = amounts[minIndex];
      amounts[minIndex] = temp;
   }
   return;
}
```

Write the function `compareAmount` according to its usage in the `if` conditional expression of the `sortAmount` function. You may use the `getCoins` function in question 13a.

Answer:

```
int compareAmount(int amount1, int amount2)
{
```

14. Elementary row operations play a very useful role in solving systems of linear equations or finding inverses of a matrix. There are three elementary row operations that can be performed. Assume that the first row of a matrix is denoted r_0.

- Interchange two rows i and j denoted $r_i \leftrightarrow r_j$
- Multiply a row i with a non-zero number s denoted $r_i = s \times r_i$.
- Reduce row i multiplied by a non-zero number s from row j denoted $r_j = r_j - s \times r_i$

Suppose `matrix` is declared as a two-dimensional array of `double` values.

(a) [**2 marks**] Write a function `swap` to interchange two rows i and j of a matrix. For example, if `matrix` represents

$$\begin{bmatrix} 0 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

then the function call `swap(matrix[0], matrix[1], 3);` performs $r_0 \leftrightarrow r_1$ where each row contains 3 values. The values of `matrix` are now

$$\begin{bmatrix} 2 & 1 & 1 \\ 0 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

Answer:

```
void swap(double x[], double y[], int n)
{
```

(b) [**2 marks**] Write a function `scale` to multiply a row i with a non-zero number s. For example, if `matrix` represents

$$\begin{bmatrix} 2 & 1 & 1 \\ 0 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

then the function call `scale(matrix[0], 3, 0.5);` performs $r_0 = 0.5 \times r_0$ where each row contains 3 values. The values of `matrix` are now

$$\begin{bmatrix} 1 & 0.5 & 0.5 \\ 0 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

Answer:

```
void scale(double x[], int n, double s)
{
```

(c) [**2 marks**] Write a function `reduce` to reduce row i multiplied by a non-zero number s from row j. For example, if `matrix` represents

$$\begin{bmatrix} 1 & 0.5 & 0.5 \\ 0 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

then the function call `reduce(matrix[2], matrix[0], 3, 1.0);` will perform $r_2 = r_2 - 1.0 \times r_0$ where each row contains 3 values. The values of `matrix` are now

$$\begin{bmatrix} 1 & 0.5 & 0.5 \\ 0 & 2 & 1 \\ 0 & 0.5 & 1.5 \end{bmatrix}$$

Answer:

```
void reduce(double x[], double y[], int n, double s)
{
```

(d) [**6 marks**] The application of elementary row operations allows us to find the inverse of a square matrix. Given the matrix $\mathbf{A}$ with a_{ij} representing the element at row i, column j, and the identity matrix $\mathbf{I}$,

$$\mathbf{A} = \begin{bmatrix} 0 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 2 \end{bmatrix} \qquad \mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

we can augment the matrix with an identity matrix to form $[\ \mathbf{A} \mid \mathbf{I}\ ]$

$$\left[\begin{array}{ccc|ccc} 0 & 2 & 1 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 2 & 0 & 0 & 1 \end{array}\right]$$

Use the elementary row operations to resolve a_{00} (the top left element) such that it is a non-zero number, and $a_{i0} = 0$ where $i \neq 0$.

$$\left[\begin{array}{ccc|ccc} 0 & 2 & 1 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 2 & 0 & 0 & 1 \end{array}\right]$$

$$\xrightarrow{r_0 \leftrightarrow r_1} \left[\begin{array}{ccc|ccc} 2 & 1 & 1 & 0 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 1 & 1 & 2 & 0 & 0 & 1 \end{array}\right]$$

$$\xrightarrow{r_2 = r_2 - 0.5 \times r_0} \left[\begin{array}{ccc|ccc} 2 & 1 & 1 & 0 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0.5 & 1.5 & 0 & -0.5 & 1 \end{array}\right]$$

You need to find an appropriate factor to perform the row reduction as shown above.

Use the elementary row operations to resolve a_{11} such that it is a non-zero number, and $a_{i1} = 0$ where $i \neq 1$.

$$\left[\begin{array}{ccc|ccc} 2 & 1 & 1 & 0 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0.5 & 1.5 & 0 & -0.5 & 1 \end{array}\right]$$

$$\xrightarrow{r_0 = r_0 - 0.5 \times r_1} \left[\begin{array}{ccc|ccc} 2 & 0 & 0.5 & -0.5 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0.5 & 1.5 & 0 & -0.5 & 1 \end{array}\right]$$

$$\xrightarrow{r_2 = r_2 - 0.25 \times r_1} \left[\begin{array}{ccc|ccc} 2 & 0 & 0.5 & -0.5 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1.25 & -0.25 & -0.5 & 1 \end{array}\right]$$

Use the elementary row operations to resolve a_{22} such that it is a non-zero number, and $a_{i2} = 0$ where $i \neq 2$.

$$\left[\begin{array}{ccc|ccc} 2 & 0 & 0.5 & -0.5 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1.25 & -0.25 & -0.5 & 1 \end{array}\right]$$

$$\xrightarrow{r_0 = r_0 - 0.4 \times r_2} \left[\begin{array}{ccc|ccc} 2 & 0 & 0 & -0.4 & 1.2 & -0.4 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1.25 & -0.25 & -0.5 & 1 \end{array}\right]$$

$$\xrightarrow{r_1 = r_1 - 0.8 \times r_2} \left[\begin{array}{ccc|ccc} 2 & 0 & 0 & -0.4 & 1.2 & -0.4 \\ 0 & 2 & 0 & 1.2 & 0.4 & -0.8 \\ 0 & 0 & 1.25 & -0.25 & -0.5 & 1 \end{array}\right]$$

Now normalize a_{00}, a_{11} and a_{22} by scaling every row.

$$\left[\begin{array}{ccc|ccc} 2 & 0 & 0 & -0.4 & 1.2 & -0.4 \\ 0 & 2 & 0 & 1.2 & 0.4 & -0.8 \\ 0 & 0 & 1.25 & -0.25 & -0.5 & 1 \end{array}\right]$$

$$\xrightarrow{r_0 = 0.5 \times r_0} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -0.2 & 0.6 & -0.2 \\ 0 & 2 & 0 & 1.2 & 0.4 & -0.8 \\ 0 & 0 & 1.25 & -0.25 & -0.5 & 1 \end{array}\right]$$

$$\xrightarrow{r_1 = 0.5 \times r_1} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -0.2 & 0.6 & -0.2 \\ 0 & 1 & 0 & 0.6 & 0.2 & -0.4 \\ 0 & 0 & 1.25 & -0.25 & -0.5 & 1 \end{array}\right]$$

$$\xrightarrow{r_2 = 0.8 \times r_2} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -0.2 & 0.6 & -0.2 \\ 0 & 1 & 0 & 0.6 & 0.2 & -0.4 \\ 0 & 0 & 1 & -0.2 & -0.4 & 0.8 \end{array}\right]$$

The inverse is simply the augmented part, i.e.

$$\begin{bmatrix} -0.2 & 0.6 & -0.2 \\ 0.6 & 0.2 & -0.4 \\ -0.2 & -0.4 & 0.8 \end{bmatrix}$$

Write the function `findInverse` that computes the inverse of any invertible square matrix `A` of size `SIZE` × `SIZE` and stores in matrix `B`. **The matrix `A` should not be modified in any way.**

Answer:

```
void findInverse(double A[][SIZE], double B[][SIZE])
{
```

— **END OF PAPER** —