NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 1 AY2013/2014

CS1010E — PROGRAMMING METHODOLOGY

Nov / Dec 2013 Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper consists of **FOURTEEN (14)** questions and comprises **TWENTY THREE(23)** 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 **80**.
6. Calculators are allowed, but not electronic dictionaries, laptops, tablets, or other computing devices.
7. Do not look at the questions until you are told to do so.
8. Please write your **Student Card number** below.

This portion is for examiner's use only.

Question	Marks	Remarks
Q11		
Q12		
Q13		
Q14		
Total		

SECTION B (4 Questions : 60 Marks)
Write your answers in the space provided.

11. [10 marks] **Goldbach's conjecture** states that "every even integer greater than 2 can be expressed as the sum of two primes". For example, the number 10 can be expressed as $3 + 7$ and $5 + 5$. Complete the following program fragment such that given an integer as input, the program outputs all **unique** sum of primes if n is an even integer. Sample runs are shown below. User input is underlined.

i.
```
Enter n: 10
3 + 7
5 + 5
```

ii.
```
Enter n: 9
Not an even number greater than two
```

Answer:

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

bool isPrime(int n); /* assume isPrime function has been defined */

int main(void)
{
   int n, i; /* You may include more variables */
```

12. [**10 marks**] Study the following program.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define SEED time(0)

void printArray(int x[], int n);
void randArray(int x[], int n);

int main(void)
{
   int x[10] = {17,4,12,1,13,20,15,6,0,12};

   srand((unsigned int)SEED);

   randArray(x,10);

   printArray(x,10);

   return 0;
}

void printArray(int x[], int n)
{
   int i;

   for (i = 0; i < n; i++)
      printf("%d ", x[i]);
   printf("\n");

   return;
}
```

We would like to define a function `randArray` to randomize the positions of the original elements of the array. For example, calling `randArray(x,10)` might result in the array `x` having the values `{12,12,4,0,1,15,17,20,6,13}`. Note that the original values are retained but assigned to random positions.

(a) [**2 marks**] Propose a method to perform the randomization.

Answer:

(b) **[8 marks]** Write the function to implement the randomization in question 12a.

Answer:

13. [**20 marks**] A random walk is performed on a square floor area made up of $n \times n$ tiles. To move from one tile to another, a step can be taken in any one of four cardinal directions, i.e. north, south, east or west. Starting from a tile located at (x, y), we would like to find out the number of steps required for a random walk to visit all tiles at least once. The `randomWalk` function is given below.

```
/*
   randomWalk function performs a complete random walk on an n-by-n
   tiled area (floor), starting from the initial tile position (x,y)
   by taking successive steps in the four cardinal directions until all
   tiles have been visited.

   The total number of steps taken is returned.

   Precondition: Pointer parameters *x and *y represent the initial
                 location where 0 <= *x < n, 0 <= *y < n.
                 In addition, floor[i][j] == 0 for all 0 <= i, j < n.

   Postcondition: The ending position is output via the pointer
                  parameters *x and *y.
*/
int randomWalk(int floor[][MAX], int n, int *x, int *y)
{
   int steps = 0;

   floor[*x][*y] = 1;

   while (!isAllVisited(floor,n))
   {
      moveOneStep(floor,n,x,y);
      steps++;
   }

   return steps;
}
```

In the following parts, you may assume that the necessary C library headers and function prototypes have been included.

(a) [**6 marks**] Complete the function `isAllVisited` to determine if the every tile of the $n \times n$ floor area have been visited at least once.

Answer:

```
/*
   isAllVisited returns true if all n-by-n tiles in floor have been
   visited, false otherwise.

   Precondition: floor[i][j] >= 0 for all 0 <= i,j < n
*/
bool isAllVisited(int floor[][MAX], int n)
{
```

(b) [8 marks] Complete the function `moveOneStep` to make a valid one step move in one of the four cardinal directions.

Answer:

```
/*
   moveOneStep makes a valid one step move from the original
   position (x,y) via one of the four cardinal directions.

   Precondition: Original position before step taken is input via
                 pointer parameters *x and *y
   Postcondition: New position after step taken is output via pointer
                  parameters *x and *y
*/
void moveOneStep(int floor[][MAX], int n, int *x, int *y)
{
```

(c) [**6 marks**] Complete the program with the `main` function such that the program requests the floor dimension n, and initial location (x, y) from the user, performs the random walk, and outputs the number of steps taken to complete the walk, as well as the ending location. You may assume that the floor dimensions will not exceed 20-by-20 tiles. A sample run is shown below. User input is underlined.

```
Enter n: 9
Enter location: 4 4
Stopped at location (8,0) using 2276 steps
```

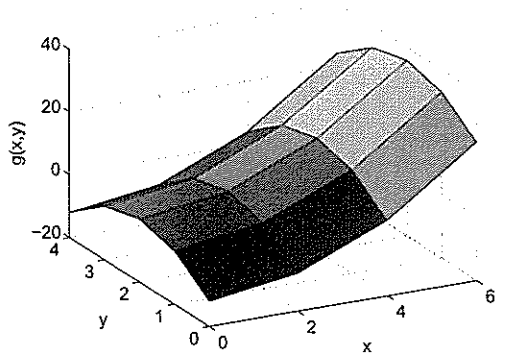
Answer:

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <time.h>

#define MAX 20
#define SEED time(0)

/* include all necessary function prototypes */
```

14. **[20 marks]** The figure below shows the plot of the function $g(x, y) = x^2 - 3(y-2)^2$ for $0 \leq x \leq 6$ and $0 \leq y \leq 4$. The step sizes of the x and y directions are set as $h_x = 2$ and $h_y = 1$ respectively. Let $g_{i,j} = g(ih_x, jh_y)$ for $i = 0, \ldots, 3$ and $j = 0, \ldots, 4$.



Study the skeleton code below and complete the accompanying tasks.

```
#include <stdio.h>
#include <math.h>
#define N 5
#define hx 2.0
#define hy 1.0

typedef struct {
   int nrow, ncol;
   double g[N][N];
} func_g;

void computeg(double, double, double, double, func_g *);
double numint2d(func_g);

int main()
{
   double x_l = 0.0, x_h = 6.0;
   double y_l = 0.0, y_h = 4.0;
   double V;
   func_g g;

   computeg(x_l,x_h,y_l,y_h,&g);
   V = numint2d(g); /* volume enclosed by surface g */
   printf("Volume = %lf\n", V);

   return 0;
}
```

(a) [10 marks] Complete the function `computeg` to compute the values of $g_{i,j}$ for all $i = 0, \ldots, 3$ and $j = 0, \ldots, 4$. Store the result in the variable g.

Answer:

```
void computeg(double x_l, double x_h,
              double y_l, double y_h, func_g *gPt)
{
```

(b) **[10 marks]** Complete the function `numint2d` to compute the volume V enclosed by the surface g for $0 \leq x \leq 6$ and $0 \leq y \leq 4$ by applying the trapezoidal rule repeatedly as follows:

Step 1. Define a 1-D array `f` as

$$f_i = h_y \left(\frac{1}{2} g_{i,0} + g_{i,1} + g_{i,2} + g_{i,3} + \frac{1}{2} g_{i,4} \right) \qquad \text{for } i = 0, 1, 2, 3.$$

f_i is the trapezoidal rule approximation of the integral of $g(ih_x, y)$ over y

Step 2. Compute

$$V = h_x \left(\frac{1}{2} f_0 + f_1 + f_2 + \frac{1}{2} f_3 \right).$$

Answer:

```
double numint2d(func_g g)
{
    int i,j;
    double V=0, f[N]={0.0};
```

— BLANK PAGE —
— END OF PAPER —