National University of Singapore

School of Computing

EXAMINATION for Semester 2 AY2010/2011

CS1010E — Programming Methodology

April/May 2011 Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This test paper contains 18 questions in two parts A and B, and comprises 11 printed pages, including this page.
2. The maximum possible mark is 40.
3. Answer *all* questions in the *space provided* in this booklet.
4. This is an *open book* examination.
5. Write your Matriculation Number below.

MATRICULATION NO:

This portion is for examiner's use only

Question	Marks	Remarks
Q1		
Q2		
Q3		
Q4		
Q5		
Q6		
Q7		
Q8		
Q9		
Q10		
Q11		
Q12		

Question	Marks	Remarks
Q13		
Q14		
Q15		
Q16		
Q17		
Q18		

Part A (Each question is worth 1 mark. Total: **15** marks)

For this part A, you are required to write down the output of the given code fragment, or write down that there is an *error*.
An error includes compiler, runtime or infinite loop error.
Your answers are to be brief and to the point.

Do not include any commentary in the answers.

Assume that the relevant `#include` pre-processor statements have been included in the code fragment as appropriate.

1.
```
int i = 5;
if (i % 3)
   printf("A");
else
   printf("B");
```

2.
```
int i = 0;
for(i = 1; i < 20; i*=2)
   printf("%d ", i);
```

3.
```
void doPrint(int, int);
int main() {
   int i = 0;
   int x = 3, y = 7;
   doPrint(y, x);
   return 0;
}

void doPrint(int x, int y) {
   printf("%d %d\n", x, y);
}
```

4.
```
int arr[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
int i = 0, j = 0;

for(i = 0; i < 3; i++)
   for(j = 0; j < 3; j++)
      printf("%d ", arr[j][i]);
```

5.
```
int main() {
   int x = 50, y = 50;
   printf("%d", func(x, y));
   return 0;
}

int func(int x, int y) {
   if(x < 0 || y < 0)
      return 0;
   else
      return func(x-1, y) + func(x, y-1);
}
```

6.
```
int i = 0, j = 0, sum = 0;
for(i = 20; i > 10; i--)
   for(j = 10; j <= 20; j++)
      sum ++;
printf("%d", sum);
```

7.
```
int i;
char arr[10] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'};
for(i = 5; i < 9; i++)
   arr[i] = arr[i+1];
for(i = 0; i < 10; i++)
   printf("%c ", arr[i]);
```

8.
```
char c = 'A', d = 5;
printf("%c %c\n", c+d, 'c'+d);
```

9.
```
char str[10] = {'A', 'B', '\0', 'D', 'E', 'F', '\0', 'H', 'I', '\0'};
printf("%s\n", str);
```

10.
```
char a[3], b[3];
a = "abc";
strcpy(b, a);
printf("%s%s", a, b);
```

11.
```
void swap(int *, int *);
int main() {
   int num1 = 5, num2 = 11, *ptr1 = &num1, *ptr2 = &num2;
   swap(ptr2, ptr1);
   printf("%d %d", num1, num2);
   return 0;
}

void swap(int *ptr1, int *ptr2) {
   int *temp = ptr1;
   ptr1 = ptr2;
   ptr2 = temp;
}
```

12.
```
int x = 5;
int *y = &x;
y = 6;
printf("%d", x);
```

13.
```
char a[5] = "abc", b[5], c[5];
strcpy(b, a);
strcpy(c, b);
b == a ? strcpy(c, "def") : strcpy(c, a);
printf("%s %s %s", a, b, c);
```

14.
```
int count = 0;
void f(int x);
int main()
{
    f(10);
    printf("%d", count);
    return 0;
}
void f(int x)
{
    count++;
    if (x > 0)
    {
        f(x/2);
        f(x/2);
    }
}
```

15.
```
struct pointer{
    int *address;
    int content;
};
void update(struct pointer p, int *new_address, int new_content)
{
    p.address = new_address;
    p.content = new_content;
}
int main()
{
    struct pointer p = {NULL, 10};
    int x = 11;
    update(p, &x, 12);
    printf("%d", p.content);
    return 0;
}
```

Part B (Total: 25 marks)

In this part, the required code fragment is only a few lines.
Comments are not necessary; if you wish to provide comments they must be restricted to a few words.
The total marks for each question is stated above each question.

16. **(5 marks)**

(a) Write a function `findtwicekey` that has three arguments: the first is an array `a` of integers, the second is an integer `n` indicating the bound the array, and the third is an integer `key`. This function is to determine if *key* is *repeated* (or appears at least twice) in the array. If so, it returns the least index i such that `a[i]` equals `key`. If not, it returns -1. [3 marks]

```
int findtwicekey(int a[], int n, int key) {






}
```

(b) This part is similar to part (a), except that no key is given. Write a function `findtwiceany` that has two arguments: the first is an array `a` of integers, and the second is an integer `n` indicating the bound the array. This function is to determine if *any* integer is repeated in the array.

If so, the function returns the least index where the repeated key resides; if not, it returns -1.

You may assume the use of the function `findtwicekey` in part (a) in the body of this function. [2 marks]

```
int findtwiceany(int a[], int n) {





}
```

17. **(10 marks)**

(a) Define a structure suitable to store information about a *book*. It should contain attributes for *title* and *author*, both strings, for the year of publication, for the ISBN number, and for the price of the book. Finally there should also be an attribute *nextedition* which is a pointer to a book structure representing the book which is the next edition of the current book. Note that a book can both be a next edition, as well as have its own next edition. Finally define a *library* as an array of books. [1 mark]

In the example below, the array `lib` is used to store a *library* of five books. Note that the book at `lib[2]` is a next edition of that at `lib[0]`, but written by a different author. The book at `lib[4]` is the next edition of that at `lib[2]`, but now the author is the same as that of `lib[0]`.

	title	author	year	isbn	price	nextedition
0	Great Gatsby	F.S. Fitzgerald	1925	4535524	25	&lib[2]
1	Jane Ayre	C Bronte	1847	9436229	20	NULL
2	Greater Gatsby	Senior Fitzgerald	2010	9846424	30	&lib[4]
3	Moby Dick	H Melville	1851	5548424	20	NULL
4	Greatest Gatsby	F.S. Fitzgerald	2011	7545314	35	NULL

(b) Assume there is a main function which populates the array of books, with global variable `numbooks` indicating the total number of books. Write function to compute the total price of books in the library by a given author. For the example above, given the author "F.S. Fitzgerald", your function should return the value 60. [4 marks]

(c) Write a function to compute total price of books by a given author including all books which are future editions of his book, even if the author of that future edition is NOT the specified author. For the example above, given the author F.S. Fitzgerald, your function should return the value 90. And given the author Senior Fitzgerald, your function should return the value 65. [5 marks]

18. **(10 marks)**
Assume you have a global two dimensional array, defined as follows:

```
#define M 100
int a[M][M];
char path[M];
int n;
```

Also assume that the cells of the array a has been populated so that the first row has one integer in the first column, the second row has two integers in the first two columns, the third row has three three integers in the first three columns, and so on. The number of rows thus populated is given by the variable n.

We wish to consider a *path* from the first row to the last or n^{th} row, and also its *cost*. A path is defined as a sequence cells such that from a from a particular cell, we can only go to a cell in the next row directly beneath the cell, or to the one situated to the right of the one beneath it. In other words, we can go either down or diagonally to the right. We start off in the topmost row and in its left most cell. Finally, the cost of a path is the *sum* of all the numbers that make up the path.

This is an example in the case n = 5.
A possible path is highlighted in bold.

6	-	-	-	-
3	**8**	-	-	-
6	**5**	4	-	-
7	1	**8**	2	-
2	4	3	**9**	7

In what follows, you are required to answer part(a) and **only one** of parts (b) and (c).
In other words, answer (a) and (b), or answer (a) and (c).
If you choose to answer (a) and (b), you may only obtain a maximum of $4+4=8$ marks.
To obtain full marks you must CORRECTLY solve (a) and (c).
The marks awarded for these parts indicate their relative difficulty.

(a) The one dimensional array path can be used to encode a path as follows. If the first step in the path is downwards, then path[0] contains 'd'; otherwise it contains 'r'. The total number of entries would then be the value of n minus one. Using the example above, the encoding 'r' 'd' 'r' 'r' indicates the path highlighted in bold.

Write a function which returns the cost of the path which is already encoded in the global array path. [4 marks]

(b) Now consider finding the path with *maximal* cost. In the example above, the path highlighted in bold in fact is such a path.

Write a *recursive* function which returns the maximal cost of the paths in a. Explain the use of formal parameters in the function, and also the actual parameters when the function is called from the main program.

Consider the following hint. From a particular cell `a[i][j]`, we can go either to `a[i + 1][j]` or to `a[i+1][j+1]`. Now, we need to maximize the sum for paths from `a[0][0]`. The maximal path value is in fact the value at `a[0][0]` plus either the maximal value of a path from `a[1][0]`, or the maximal value of a path from `a[1][1]`. In general, the maximal path value at a particular cell equals the value at that cell plus the maximal path values of cells reachable from it. [4 marks]

(c) Note that your recursive function above in (b) may not be efficient because there are an exponential number of paths. (For example, if $n = 6$ there are 32 paths to consider.)

Now write a *nonrecursive* function which returns the maximal cost of the paths. As above, this function reads the global array `a` and the array bound `n`.

Consider the following hint. Declare a new two dimensional array `max[M][M]` which stores in `max[i][j]` the maximal cost of a path from the start cell to this cell. Populate this array row by row so that the values in row i can be easily obtained by the prevsiouly computed values in row $i-1$. [6 marks]

END of PAPER