NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

MID-SEMESTER TEST
AY2017/8 Semester 1

CS1010 PROGRAMMING METHODOLOGY

6 October 2017 Time Allowed: **1 hour 30 minutes**

INSTRUCTIONS

1. This question paper contains **ELEVEN** (11**)** questions and comprises **TEN** (10**)** printed pages, including this page.
2. An **ANSWER SHEET** is provided for you to write the answers. It comprises **TWO (2)** printed pages.
3. Answer **ALL** questions within the space provided on the **Answer Sheet**.
4. Maximum score is **30 marks**.
5. This is an **OPEN BOOK** test.
6. Electronic devices, including but not limited to laptop, electronic dictionary and calculator, are **NOT** allowed.
7. Switch off/silence your mobile phone and keep it out of view.
8. Write your **DISUCSSION GROUP NUMBER** and **STUDENT NUMBER** on the **Answer Sheet** using **A PEN**.
9. You may use pencil to write your answers, but please write legibly. Marks will be deducted for illegible or untidy answers.
10. Submit only the **Answer Sheet** at the end of the test. You may keep the question paper.

——— END OF INSTRUCTIONS ———

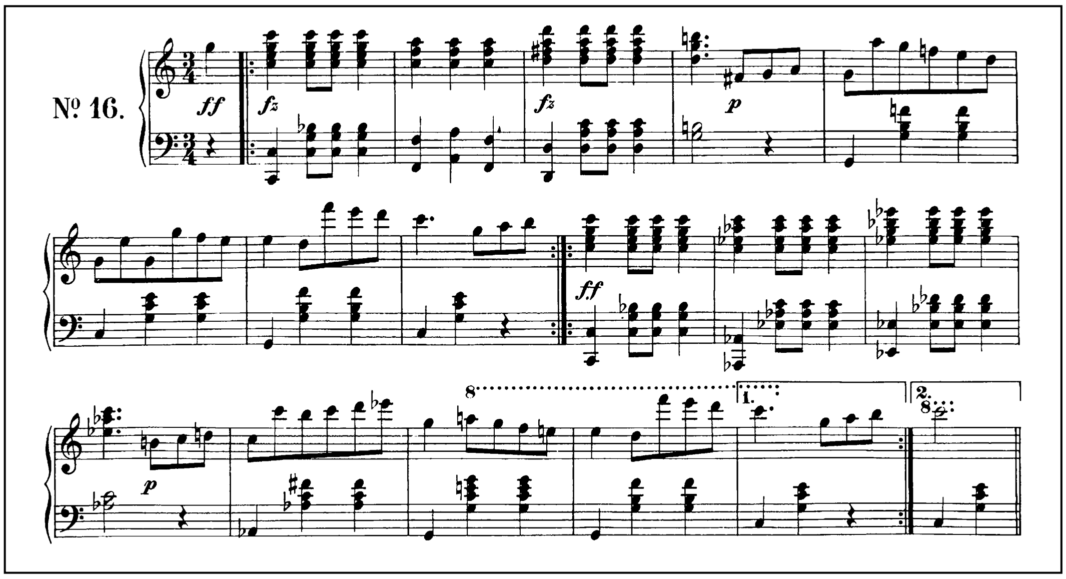
Question 0 is a bonus question that is worth one mark; its mark will only be added if you get it right and the total mark you scored is less than 30.

0. [Bonus] In a certain lecture, Aaron Tan tore a 'dictionary' repeatedly. What was he demonstrating?

   A. Binary Search
   B. Bisection Method
   C. Bubble Sort
   D. Computational Thinking
   E. He wasn't demonstrating anything; he just went crazy.

Questions 1 to 5: Each multiple-choice question has only one correct answer. Write your answers in the boxes provided on the **Answer Sheet**. Two marks are awarded for each correct answer and no penalty for wrong answer.

1. What main concept does musical notation (an example of a musical score using such a notation is shown below) exemplify?



   A. Decomposition
   B. Pattern recognition
   C. Abstraction
   D. Algorithm design
   E. None of the above

2. Given the following Bubble Sort function:

```
void bubbleSort(int arr[], int size) {
   int i, limit, temp;

   for (limit = size-2; limit >= 0; limit--) {
      for (i=0; i<=limit; i++) {
         if (arr[i] > arr[i+1]) {
            temp = arr[i];
            arr[i] = arr[i+1];
            arr[i+1] = temp;
         }
      }
   }
}
```

If the integer array *values* originally contains { 19, 20, 34, 71, 5, 88, 12, 3 }, what are the elements in *values* after three passes of Bubble Sort in the call bubbleSort(*values*, 8)?

A. 3 5 12 19 20 34 71 88
B. 19 20 5 34 12 3 71 88
C. 5 19 20 12 3 34 71 88
D. 19 5 20 12 3 34 71 88
E. None of the above

3. In a particular physical fitness proficiency test, points are awarded depending on the number of sit-ups performed according to this table:

Number of sit-ups	<30	30-34	35-39	40-44	≥45
Points awarded	0	1	2	3	4

Which of the following functions computes the points correctly?

(i)
```
// Precond: situp >= 0
int compute_points(int situp) {
  int points = 0;

  switch (situp) {
    case (situp>29): points = 1; break;
    case (situp>34): points = 2; break;
    case (situp>39): points = 3; break;
    case (situp>44): points = 4; break;
  }
  return points;
}
```

(ii)
```
// Precond: situp >= 0
int compute_points(int situp) {
  int points = 0;

  switch ((situp>29)+(situp>34)+(situp>39)+(situp>44)) {
    case 1: points++; break;
    case 2: points++; break;
    case 3: points++; break;
    case 4: points++; break;
  }
  return points;
}
```

(iii)
```
// Precond: situp >= 0
int compute_points(int situp) {
  int points = 0;

  switch ((situp>29)+(situp>34)+(situp>39)+(situp>44)) {
    case 4: points++;
    case 3: points++;
    case 2: points++;
    case 1: points++;
  }
  return points;
}
```

A. None of (i), (ii) and (iii)
B. Only (ii)
C. Only (iii)
D. Only (i) and (iii)
E. Only (ii) and (iii)

4. Study the following function carefully:

```
int binarySearch(int arr[], int size, int key) {
   int left = 0, right = size -1, mid, index = -1;

   while (left <= right && index == -1) {
      mid = (left + right)/2;
      if (arr[mid] == key)
         index = mid;
      else {
         if (arr[mid] > key)
            right = mid;
         else
            left = mid;
      }
   }
   return index;
}
```

The above function is called with the array *values* declared as follows:

```
int values[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
```

Which of the following statements is/are true?

i. Calling **binarySearch(values, 9, 1)** returns the value of 0.

ii. Calling **binarySearch(values, 9, 1)** results in an infinite loop.

iii. Calling **binarySearch(values, 9, 9)** returns the value of 8.

iv. Calling **binarySearch(values, 9, 9)** results in an infinite loop.

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

5. Study the following function carefully. MAX is an integer constant defined earlier in the program.

```
int mystery(int mat[][MAX], int size) {
   int r, c;

   for (r = 1; r < size; r++) {
      for (c = 0; c < r; c++) {
         if (mat[r][c] < mat[c][r]) {
            return 1;
         }
      }
   }
   return 0;
}
```

The function is called by passing each of the following square matrices with *size* = 5. Which of the following matrices will result in the function returning the value 0?

(i)
$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}$$

(ii)
$$\begin{bmatrix} 25 & 24 & 23 & 22 & 21 \\ 20 & 19 & 18 & 17 & 16 \\ 15 & 14 & 13 & 12 & 11 \\ 10 & 9 & 8 & 7 & 6 \\ 5 & 4 & 3 & 2 & 1 \end{bmatrix}$$

(iii)
$$\begin{bmatrix} 3 & 5 & 19 & 6 & 13 \\ 12 & 0 & 0 & 11 & 9 \\ 20 & 23 & 19 & 15 & 20 \\ 8 & 15 & 21 & 11 & 8 \\ 20 & 12 & 39 & 6 & 22 \end{bmatrix}$$

(iv)
$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 9 & 1 & 5 & 3 \\ 6 & 5 & 5 & 6 & 1 \\ 4 & 8 & 7 & 0 & 8 \\ 5 & 5 & 2 & 8 & 9 \end{bmatrix}$$

A. Only (i)
B. Only (i), (iii) and (iv)
C. Only (iii) and (iv)
D. Only (ii) and (iii)
E. Only (i) and (iv)

Questions 6 to 11: Write your answers in the space provided on the **Answer Sheet**. You may write in pencil. You are to write legibly or marks might be deducted.

6. The *Number Hive Puzzle* is a logic puzzle consisting of a block of hexagons, with different areas marked out using thicker lines. There are two rules that must hold of a completed block.

   - Each area must contain the numbers from 1 up to the number of hexagons in the area. For example, in Figure 1(a) below, the topmost area contains 4 hexagons so they must be filled with 1, 2, 3 and 4 with no repeated numbers.
   - No number can be next to the same number in any direction along a shared edge.

   Figure 1(a) below shows an example and Figure 1(b) is the solved puzzle.

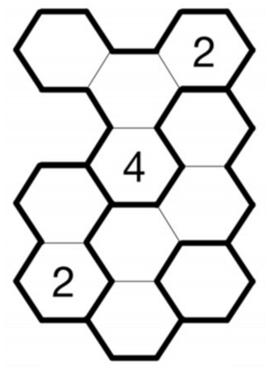


Figure 1(a)

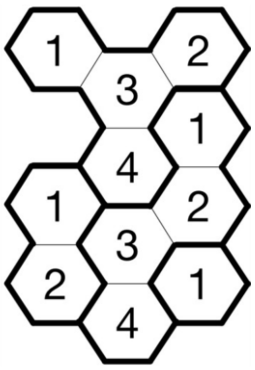


Figure 1(b)

Solve the Number Hive Puzzle in Figure 2 below. [2 marks]

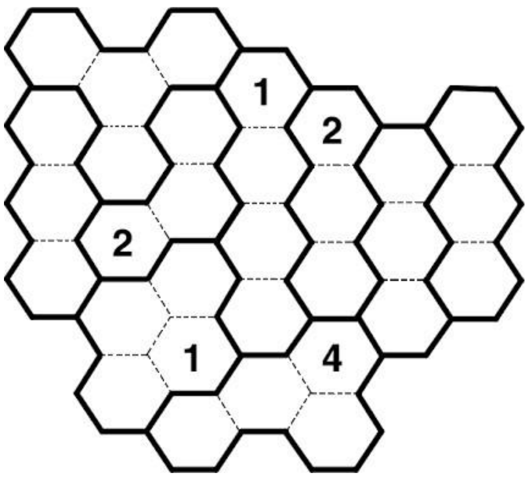


Figure 2

7. Describe what the function below does to the array *list*. [2 marks]

```
void arrange(float list[], int size) {
   int i;
   float temp;

   for (i = 0; i < size; i++) {
      temp = list[i];
      list[i] = list[size - 1 - i];
      list[size - 1 - i] = temp;
   }
}
```

8. Given the following program, what is the output? [2 marks]

```
#include <stdio.h>

int f1(int, int);
int f2(int);

int main(void) {
   int a = 100, b = 200;

   a = f1(b, f2(a));
   printf("%d %d\n", a, b);

   return 0;
}

int f1(int a, int b) {
   a = f2(b);
   return a + b;
}

int f2(int a) {
   return a/2;
}
```

9. Given the following program, what is the output? [2 marks]

```
#include <stdio.h>
#define SIZE 8

int main(void) {
   int arr[SIZE] = { 1, 2, 3, 4, 5, 6, 7, 0 };
   int i, j;

   for (i = 0; i < SIZE - 1; i++) {
      for (j = i+1; j < SIZE; j++) {
          arr[i] = (arr[i] + arr[arr[j]])%SIZE;
      }
   }

   for (i = 0; i < SIZE; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");

   return 0;
}
```

10. Study the following function carefully:

```
// Precond: size >=0
void process_array(int arr[], int size) {
   int i, j, sum;

   for (i=size-1; i>=0; i--) {
      sum = 0;
      for (j=0; j<=i; j++) {
         sum = sum + arr[j];
      }
      arr[i] = sum;
   }
}
```

The function above was poorly written. Replace it with a simpler code so that it performs the same task. Your code will be assessed not only for its correctness but also its elegance. [4 marks]

11. [8 marks]
A stock market company tracks the daily prices of stocks and wants to find out if a stock is *favourable*. A favourable stock is one where the price may rise or fall, but each time it falls, subsequent prices should not fall below that price in future. For example, if on some day the price has fallen to $x, then on subsequent days the price should never fall below $x. Furthermore, the price of the stock should not fall below its first day's price.

(This question is to test you on understanding problem statement, so no examples will be given. Hint: Think of a way to make it easy to answer part (a) below.)

You are given an integer array *prices* that represents the daily prices of a particular stock and *size* is the number of prices in the array. Assuming that the array has at least one price, do the following:

a. Write "true" or "false" for each of the following three test cases, where "true" means the stock is favourable and "false" means it is not. [3 marks]

    i. int prices[] = { 100, 200, 150, 130, 300 };

    ii. int prices[] = { 1, 10, 2, 8 , 2, 7, 2, 5 };

    iii. int prices[] = { 10, 20, 12, 30, 15, 35, 12, 40 };

b. Write a function **favourable(int arr[], int size)** that returns 1 if the stock is favourable, or 0 otherwise. You may assume that the stock contains at least one price. Your code will be assessed not only for its correctness but also its elegance. [5 marks]

——— END OF PAPER ———