TIC1001—Introduction to Coputing and Programming
National University of Singapore
School of Continuing and Lifelong Education

Midterm Test

5 October 2017 **Time allowed:** 1 hour 30 mins

Student No:	**A**								

Instructions (please read carefully):

1. Write down your **student number** on the question paper. DO NOT WRITE YOUR NAME ON THE ANSWER SET!
2. This is **an open-sheet test**. You are allowed to bring one A4 sheet of notes (written or printed on both sides).
3. This paper comprises **SIX (6) questions** and **FOURTEEN (14) pages**. The time allowed for solving this test is **1 hour 30 mins**.
4. The maximum score of this test is **100 marks**. The marks for each question is given in square brackets beside the question number.
5. Note that the marks and order of the questions do not necessarily correspond to the level of difficulty of the question.
6. All questions must be answered correctly for the maximum score to be attained.
7. The pages marked "scratch paper" in the question set may be used as scratch paper.
8. You are allowed to un-staple the sheets while you solve the questions. Please make sure you staple them back in the right order at the end of the test.
9. You are allowed to use pencils, ball-pens or fountain pens, as you like as long as it is legible (no red color, please).

EXAMINER'S USE ONLY

Question	Marks
Q1	/ 20
Q2	/ 28
Q3	/ 18
Q4	/ 10
Q5	/ 8
Q6	/ 16
Total	/ 100

ALL THE BEST!

This page is intentionally left blank.

It may be used as scratch paper.

Question 1: C Expressions [20 marks]

There are several parts to this question which are to be answered independently and separately. Each part consists of a fragment of C code. Write the exact output produced by the code in **the answer box**. If an error occurs, or it enters an infinite loop, state and explain why.

You may show workings **outside the answer box** in the space beside the code. Partial marks may be awarded for workings if the final answer is wrong.

Assume that all appropriate preprocessor directives e.g., `#include <stdio.h>`, etc. have already been defined.

A. [2 marks]

```
int square(double x) {
    return x * x;
}
printf("%d", square(1.5));
```

B. [2 marks]

```
int x = 2;
int square(int x) {
    return x * x;
}
printf("%d", square(square(3)));
```

C. [2 marks]

```
int x = 5;
int y = 3;
double foo(int x, int y) {
    return x/y;
}
printf("%f", foo(y, x));
```

D. [2 marks]

```
int foo(int x, double y) {
    printf("%.2f ", y/x);
    return y/x;
}
foo(foo(4, 10), 15);
```

E. [3 marks]

```
int i;
for (i = 0; i < 10; ++i) {
    if (i % 2 == 0) { ++i; }
    if (i % 3 == 0) { --i; }
    if (i % 5 == 0) { break; }
}
printf("%d", i);
```

F. [3 marks]

```
int a = 3;
int f(int a) {
    a = 7;
    return a;
}
int g(int a) {
    return f(a) + a;
}
printf("%d", g(a+2) + f(a-1));
```

G. [3 marks]

```
int i = 20, j = 0;
while (i >= j) {
    --i;
    if (i < 3*j) { continue; }
    j += 2;
    printf("%d ", j);
}
printf("%d", i);
```

H. [3 marks]

```
int x = 0;
int f(int *x) {
    if (*x > 0) {
        *x = -*x;
    } else {
        *x += 2;
    }
    return *x;
}
x = f(&x) + f(&x);
printf("%d", x);
```

Question 2: Traffic Lights [28 marks]

A traffic light has three states: green, amber and red. We can use an integer to represent the state of a traffic light using the values 0, 1, and 2, respectively.

A. Implement the function `void display(int light)` that takes as input a traffic light represented as an integer. The function prints the string `"Green"`, `"Amber"` or `"Red"` based on the value of the light. [6 marks]

```
void display(int light) {









}
```

B. In Singapore, our traffic lights cycles from green, to amber, to red, then back to green. The function `change` takes as input a traffic light (which is an integer) and changes it to the next state.

Decide on the appropriate type and inputs of the function `change` and provide an implementation of it in C. [6 marks]

C. We can consider one traffic light cycle to be a change from its current state to another, until it returns back to its starting state.

Suppose that a function `void wait(int secs)` has been defined which pauses the program execution for the given number of seconds. Implement a function `cycle` that takes as input a traffic light, and the timings between the states. The function changes and display the state of the traffic light with the appropriate pauses between the states for one cycle.

For example, if the light is currently green, the call `cycle(&light, 5, 20, 20)` will result in the following being printed (commented lines are not printed, but explains the behaviour):

```
Amber
// pause of 5 seconds
Red
// pause of 20 seconds
Green
// pause of 20 seconds
```

Provide an implementation for the function `cycle`. Note that the value of the light has to change, and you should reuse the functions defined in the previous parts. [8 marks]

D. A particular traffic light cycles with the following timings:

Cycle	Green	Amber	Red	Repeat
1	58 s	2 s	60 s	10 times
2	38 s	2 s	90 s	5 times
3	88 s	2 s	40 s	5 times

Basically, cycle 1 will repeat 10 times, followed by cycle 2 for 5 times then cycle 3 for 5 times. Then the entire cycle will repeat from cycle 1 again.

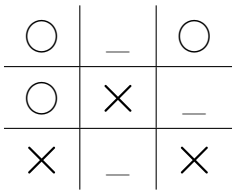
Complete the following code such that the program will print the state of this particular traffic light when it changes with the required pause between changes. E.g., starting at cycle 1, `"Green"` will be printed followed by `"Amber"` after pausing for 58 s, and `"Red"` after a pause of 2 s. The program should continue running indefinately, displaying the changes according to the stated schedule.

You are to reuse the functions defined in the previous parts. [8 marks]

```
int main(void) {
    int light = 0;









}
```

Question 3: Tic Tac Toe [18 marks]

Suppose we want to store the board state of a **tic-tac-toe** game, for example:



We can use numbers 0, 1 and 2 to represents the markers _ (empty space), ◯ and ✕ respectively. Then, going by rows from top to bottom, the whole board can be represented by a 9-digit integer T. For example, $T = 101120202$ represents the game board above.

A. Draw the tic-tac-toe board represented by the $T = 202012111$. [2 marks]

B. What is the smallest and largest possible value for T? There is no need to consider whether the board can be the result of a actual game as long as each location has a valid marker. [2 marks]

C. The representation is not very efficient in terms of storage space (i.e. it uses more bits then absolutely necessary). Briefly explain why. [2 marks]

D. Use your understanding of **number base**, briefly suggest a more efficient way to represent the game state as a **single number**.

Use your scheme to represent the example game board given in the main question. You can leave the representation in any number base as long as you indicate the base clearly. [4 marks]

E. It turns out that the board represented in Part A is not a valid game. In a game of tic-tac-toe, the two players represented by ◯ and ✕, take alternating turns to fill an empty space with their mark. Assuming that ✕ always goes first, there can never be more ◯ than ✕ and there can at most be 1 more ✕ than ◯ in a valid game board.

Implement the function `is_valid(int board)` which takes as input a game board using our representation given at the start of this question. The function returns **`true`** if the game board is valid based on the number of markers, and **`false`** otherwise.

Examples:

Function call	Return value
`is_valid(202012111)`	`false`
`is_valid(111222000)`	`true`
`is_valid(210120020)`	`true`

[8 marks]

```
#include <stdbool.h>

bool is_valid(int board) {









}
```

Question 4: Computer Organization [10 marks]

Consider the following "assembly" code:

```
1 load r1, 10      # r1 initialized to 10
2 load r2, 0       # r2 initialized to 0
3 load r3, 1       # r3 initialized to 1
4
5 add  r4, r2, r3  # r4 = r2 + r3
6 add  r2, r3, 0
7 add  r3, r4, 0
8 if (r4 < r1), goto line 5
9 <ending here>
```

A. What are the values stored in registers $r2$, $r3$ and $r4$ when the code reaches line 8 **for the first time**? [3 marks]

r2 =	r3 =	r4 =

B. What are the values stored in registers $r2$, $r3$ and $r4$ when the code reaches line 9 **eventually**? [3 marks]

r2 =	r3 =	r4 =

C. Write a C code with similar behaviour as the "assembly" code above. For simplicity, you can just use the register name as variable names. [4 marks]

```
int r1, r2, r3, r4;   //assume you have these variables
```

Question 5: IP Addressing [8 marks]

Suppose we have an IP address:

137	54	44	123
1000 1001	0011 0110	0010 1100	0111 1011

Give the **largest** CIDR block size, i.e. the maximum number of bits in prefix, so that the following IP address are in the **same block**.

A. `137.54.44.244` [2 marks]

CIDR Block Size =

B. `137.54.45.123` [2 marks]

CIDR Block Size =

C. `137.48.44.123` [4 marks]

CIDR Block Size =

Question 6: Type of Errors [16 marks]

The following C function attempts to reverse the digits in the input parameter n and returns the value via the pointer parameters r:

```
1  void reverse(int n, int *r)
2  {
3      int result = 0;
4
5      while (n > 10 ) {
6          result = (10 * result) + (n % 10);
7          n = n / 10;
8
9      r = result;
10
11     return r;
12 }
```

However, there are a number of errors in the implementation. For each error, indicate the line number, the type of error and the corrected statement. You should consider each mistake separately from others. [16 marks]

Line No	Type of Error	Correction

— E N D O F Q U E S T I O N S —

Scratch Paper

— E N D O F P A P E R —