NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING
MIDTERM ASSESSMENT FOR
Semester 1 AY2018/2019

CS1010 Programming Methodology

October 2018 Time Allowed 90 Minutes

INSTRUCTIONS TO CANDIDATES

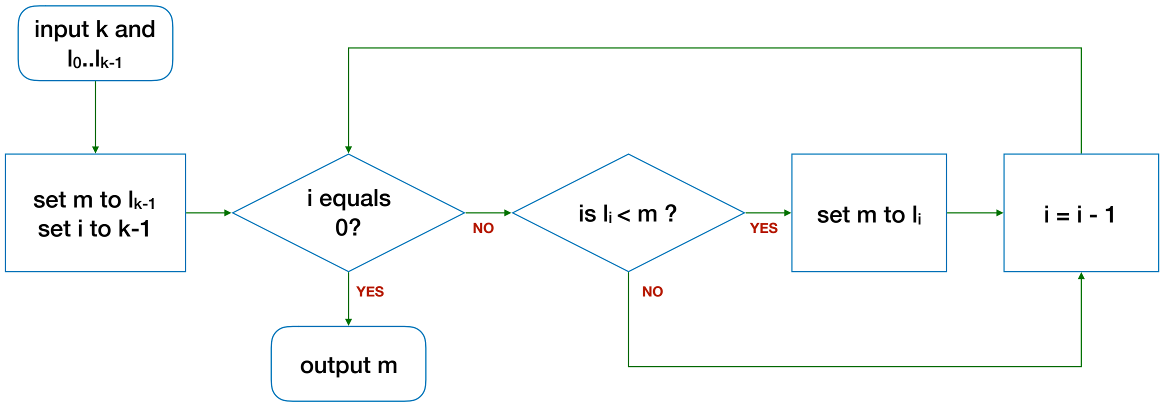
1. This assessment paper contains 15 questions and comprises 10 printed pages, including this page.
2. Write all your answers in the answer sheet provided.
3. The total marks for this assessment is 60. Answer **ALL** questions.
4. This is an **OPEN BOOK** assessment.
5. You can assume that in all the code given, no overflow nor underflow will occur during execution. In addition, you can assume that all given inputs are valid and fits within the specified variable type.
6. State any additional assumption that you make.
7. Please write your student number only. Do not write your name.

Part I

Multiple Choice Questions (30 points)

- For each of the questions below, **write your answer in the corresponding answer box on the answer sheet.** Each question is worth 3 points.

1. (3 points) The flowchart below shows an algorithm that takes in a list $L = l_0, l_1, l_2, ..., l_{k-1}$ and the number of elements k as input.



What is the output if the input list L is 1, 2, 7, 4 and k is 4?

A. 1

B. 2

C. 7

D. 4

E. No output

Write X in the answer box if none of the choices above is correct.

Solution: B. The algorithm starts from 4, and checks through 7 and 2, for the minimum. It stops when $i == 0$ so there is no opportunity to check and compare l_0. So the minimum returned is 2.

2. (3 points) Consider the function `square` as defined below.

```
long square(long x)
{
  return x*x;
}
```

What is the return value of the expression `square(square(square(2)))` ?

A. 4

B. 6

C. 8

D. 16

E. 64

Write `X` in the answer box if none of the choices above is correct.

Solution: X.

`square(square(square(2)))` is the same as `square(square(4))`, which is `square(16)`, which is 256.

3. (3 points) What would be the value of variable `x` after executing the C statement below?

```
double x = (1/2) * (2/3) * (3/4);
```

   A. 0.00
   B. 0.25
   C. 1.00
   D. 1.50
   E. 2.00

   Write `X` in the answer box if none of the choices above is correct.

Solution: A. The expression `1/2`, `2/3`, `3/4`, all return `0`.

4. (3 points) Consider the function `f` as defined below.

```
bool f() {
  if (a) {
    return true;
  } else if (b) {
    if (c) {
      return false;
    }
  }
  return true;
}
```

   The body of the function `f` is equivalent to which of the following logical expression?

   A. `return (!a && b && c);`
   B. `return (!a && b || c);`
   C. `return (a && !(b && c));`
   D. `return (a && !b && !c);`
   E. `return (a || !(b && c));`

   Write `X` in the answer box if none of the choices above is true.

Solution: E. The function returns `false` if `a` is false and `b && c` is true. So it returns false if `(!a && (b && c))`. Taking the de Morgan's law, we have E.

5. (3 points) Consider the `mystery` function defined below.

```
long mystery(long x) {
  if (x == 1) {
    return 1;
  }
  return 1 + mystery(x/2);
}
```

What is the return value of `mystery(10)` ?

A. `6`

B. `5`

C. `4`

D. `3`

E. `2`

Write `X` in the answer box if none of the choices above is true.

Solution: C.

`mystery(10)` is the same as `1 + mystery(5)`, which is `1 + (1 + mystery(2))`, which is `1 + 1 + (1 + mystery(1))`. This gives us 4.

6. (3 points) Consider the `if-else` statement below.

```
if (z <= 100) {
  z = z/2;
  if (y > z) {
    z = y;
  } else {
    y += 10;
    // Line B
  }
}
```

What can we assert at Line B?

A. `y >= 10`

B. `y <= 20`

C. `y <= 50`

D. `y <= 60`

E. `y >= 100`

Write `X` in the answer box if none of the choices above is true.

Solution: After the line `z = z/2`, we know `z <= 50`. Line B is in the false block of `y > z`, so we know that `y <= z`, i.e., `y <= 50`. After the line `y += 10`, we know that `y <= 60`.

D.

7. (3 points) The marriage law in Singapore states that a person can get married if and only if either (i) he or she is at least 21 years old, or (ii) he or she is at least 16 years old and have the consent of the parents.

Suppose you want to write a function `cannot_get_married` that takes in a `long` variable `age`, and a `bool` variable `consent` (which is set to `true` if the person has the consent of the parents and `false` otherwise). Your function should returns `true` if the person is NOT allowed to get married under the law of Singapore, and returns `false` otherwise.

Which of the following return statement of your function is the correct one?

A. `return (age < 16) && !consent;`

B. `return (age < 16) || !consent;`

C. `return (age >= 16 && age < 21) && !consent;`

D. `return (age < 21) || (age < 16 && !consent);`

E. `return (age < 21) && (age < 16 || !consent);`

Write `X` in the answer box if none of the choices above is true.

Solution: Let's look at the expression for allowing marriage first:

```
(age >= 21) || (age >= 16 && consent)
```

Now apply De Morgan's Law:

```
(age < 21) && !(age >= 16 && consent)(age < 21) && (!(age >= 16) || !consent)
(age < 21) && (age < 16 || !consent)
```

E.

8. (3 points) Which of the following function or functions correctly compute the sum of all integers between `i` and `j`, inclusive? You can assume that `j >= i`.

```
long foo(long i, long j) {
  long sum = i;
  long x = i;
  while (x < j) {
    x += 1;
    sum += x;
  }
  return sum;
}

long bar(long i, long j) {
  long sum = 0;
  long x = i;
  do {
    sum += x;
    x += 1;
  } while (x <= j);
  return sum;
}

long qux(long i, long j) {
  long sum = 0;
  for (long x = j; x >= i; x -= 1) {
    sum += x;
  }
  return sum;
}
```

A. Only `foo`
B. Only `qux`
C. Only `bar` and `qux`
D. Only `foo` and `qux`
E. `foo`, `bar` and `qux`

Write `X` in the answer box if none of the combinations is correct.

Solution: E.

9. (3 points) Consider the ‘for‘ loop below. Line A in `update` function is left blank.

```
#include "cs1010.h"

long update(long x)
{
  // Line A
}

int main()
{
  for (long x = 1; x <= 10; x = update(x)) {
    cs1010_println_string("Wakanda");
  }
}
```

Which of the following statement, if inserted into `Line A`, would cause the program to go into an infinite loop and print `Wakanda` forever?

(i) `return x*x;`

(ii) `return x + 0.0001;`

(iii) `return x;`

A. (i) only

B. (iii) only

C. (i) and (iii) only

D. (ii) and (iii) only

E. (i), (ii), and (iii)

Write `X` in the answer box if none of the combinations is correct.

Solution: E.

10. (3 points) Consider the while loop below.

```
do {
  if (b) {
    // { X }
  } else if (c) {
    // { Y }
  }
} while (a);
// {  ??  }
```

`a`, `b` `c`, `X` and `Y` are all logical expressions. At the end of the true-block for expression `b`, we can assert that `X` is true. At the end of true-block for expression `c`, we can assert that `Y` is true.

What can we assert on the line marked `??` ?

A. `a && X && Y`

B. `a && (X || Y)`

C. `!a && (X || Y)`

D. `!a && ((X || Y) || (!b && !c))`

E. `!a && ((X || Y) && (!b && !c))`

F. `!a && ((X || Y) && (b || c))`

Write `X` in the answer box if none of the choices above is correct.

Solution: We know that after the while loop, `!a` must be true. Two things can happened in the loop. If we entered the 'if-else' branch, then `(X || Y)` must be true. Otherwise, then `!b && !c` must be true. So D.

Part II

Short Questions (30 points)

Answer all questions in the space provided on the answer sheet. Be succinct and write neatly.

11. (6 points) **Algorithm.** Design an algorithm to find the second largest integer from a given list $L = l_0, l_1, ..., l_{k-1}$ with k elements. You can assume that $k \geq 2$ and the list L does not contain any repetition (i.e., the same number does not appear twice).

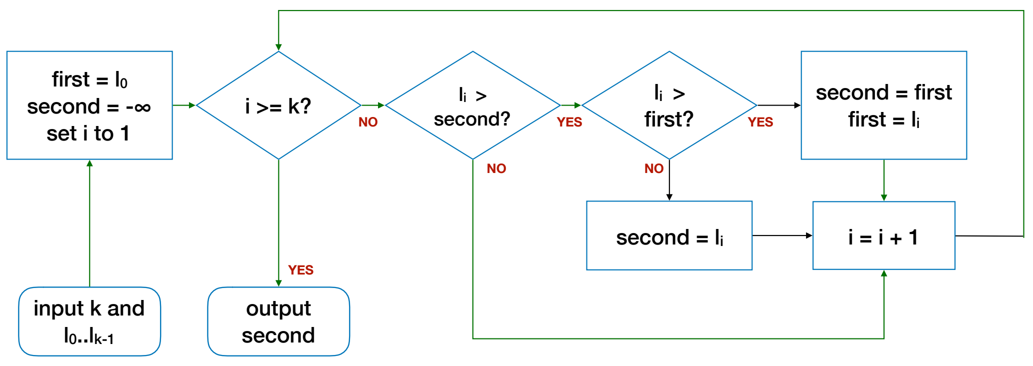
An easy but inefficient solution is to scan through the list twice: First, scan through the list to find the maximum integer m, using the algorithm from the class. Then, scan through the list again, finding the maximum integer that is not m.

You should do better than the algorithm above. Your algorithm must scan through the list once only.

Describe your algorithm using a flowchart.

Solution:

This is a variation of the problem we have seen many times in CS1010 – finding the maximum from the list. Surprisingly many are stumped by this problem. The key idea to this solution is to keep track of two variables – the largest so far and the second largest so far, and update them accordingly as we scan through the array. Here is the flowchart.



Some common errors:

- Failing to initialize `first` and `second` properly. Some of you initialized `second` to one of the variables in the list (usually l_0 or $_l$). If this variable happens to be the largest in the list, then your code would fail. Some of you initialized `second` to 0. You code would not work for negative numbers (you have seen at the beginning of the semester that the right value to initialize to is $-\infty$. You get -1 mark for incorrect initialization.

- Failing to compare l_i with `second`. This is a more serious bug as this is important for updating the second largest element. We deducted 2 marks for this bug.

- Failing to update `second` if we find $l_i > first$. Those committed this bug either forgot to set `second` to `first` at all, or set `second` to `first` after setting `first` to l_i.

For other misc bugs, we deducted 1 mark for each of them (e.g., incorrect terminating condition, setting l_i to `max` instead of the other way around, etc).

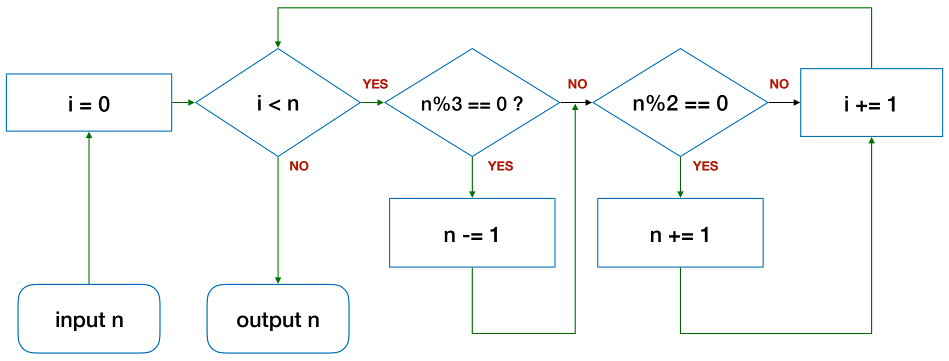
A solution that simply returns maximum or uses only one variable to scan for the second largest element received 0 marks. Similarly, a solution that scans through the array more than once received 0 marks.

Some of you did not follow the convention of drawing flowcharts – using diamonds for branches, rectangular for statements. We decided not to penalize such mistakes this time (but we may in the final assessment). Those who drew a flowchart without any arrow gets 1 marks off. It is almost impossible to trace the chart without knowing the direction.

12. (6 points) **Flowchart.** Draw the corresponding flowcharts for the code below:

```
long foo(long n)
{
  for (long i = 0; i < n; i += 1) {
    if (n % 3 == 0) {
      n -= 1;
    }
    if (n % 2 == 0) {
      n += 1;
    }
  }
  return n;
}
```

Solution: Here is the flowchart:



We deducted 1 mark for every bug in the flowchart, except the following: -0.5 for missing input block and -0.5 for extra inputs (some students passed in $l_0, l_1, ..l_{k-1}$ into this function, for some unknown reason).

13. (6 points) **Loops.** Trace through the code below. What are the values of `i`, `j`, `k` at Line B?

```
long i = 1;
long j = 0;
long k = 0;
while (i <= 4) {
  i *= 2;
  while (j < i) {
    k += 1;
    j += 1;
  }
  j = 0;
}
// Line B
```

Solution: This question assesses if students are able to read and trace through non-trivial nested loops. 2 marks per correct value given.

`i` is 8, `j` is 0, `k` is 14.

14. (6 points) **Invariant.** Consider the function given below.

```
long foo(long n, long k) {
  long x = n;
  long y = 0;
  while (x >= k) {
    // Line C
    y += 1;
    x -= k;
    // Line D
  }
  // Line E
}
```

(a) (1 point) What are the values of `x` and `y` on Line E if `n` is 3 and `k` is 5?

(b) (1 point) What are the values of `x` and `y` on Line E if `n` is 10 and `k` is 2?

(c) (1 point) What are the values of `x` and `y` on Line E if `n` is 8 and `k` is 3?

(d) (1 point) Write down a loop invariant for the while loop in the form of `x == ??` (you are supposed to replace `??` with an arithmetic expression). Note that you should not write trivial invariants such as `x == x`.

(e) (2 points) Explain why the loop invariant is true at Line D assuming that it is true at Line C.

Solution:

(a) `x` is 3, `y` is 0

(b) `x` is 0, `y` is 5

(c) `x` is 2, `y` is 2

(d) `x == n - y*k`

(e) At Line C, assuming `x == n - y*k`, after `y += 1`, we have `x == n - (y-1)*k`. In the next line, we decrement `x -= k`, so `x + k == n - (y-1)*k`, simplifying the right-hand side, we have `x == n - y*k` again.

Part (a)-(c) is meant to be give-away marks to help you see the pattern in the code and understand what the code is doing.

Part (d) is where the real challenge begins: you were asked to write a loop invariant here. Some common wrong answers are:

- `x == n % k`. Remember that an invariant is an assertion that is true at the beginning of the loop, during the loop, and at the end of the loop. This assertion, however, is only at the end of the loop.
- `x == x - yk`. This is wrong since it implies that `yk` is 0.

Even though this part is only 1 mark, it affects Part (e) below.

Part (e) asked you to argue why the invariant is true at Line D assuming it is true at Line C. If your invariant is wrong, then most likely you get zero for this question. We expect you to put forth a step-by-step argument of how the four variables related to each other as their value changes from Line C to Line D, as we did in class. Some of you made an argument for the correctness of the invariant from the overall behavior of the program. Depending on how detailed your argument is, you may get anywhere between 0 - 2 marks. An example of such argument is this: y counts how many times we loop. Every loop we deduct k from x, so x == n - yk). Such argument, however, is harder to make. We have to argue that x begins with n, there is no other code that modifies y and x, etc. The scope of the argument suddenly expands from the two lines between Line C and Line D to the whole function.

15. (6 points) **Recursion.** Write a *recursive* function `has8` in C that, given an positive integer number, returns `true` if the digit 8 appears somewhere in the number, returns `false` otherwise. Your code should consist of a single `return` statement and appropriate use short-circuiting to avoid unnecessary checks.

    Note: An iterative solution using loops would receive 0 marks.

```
bool has8(long n)
{
   return ??;
}
```

Solution: `return (x % 10 == 8) || (x > 10 && has8(x/10));`

A wide spread error is missing the terminating condition `x > 10`. Without this, your code would run forever. We deducted two marks if you miss this.

This question also assesses if you know about short-circuiting and know how to use it. Exploiting short-circuiting, we should make the recursive calls `has8(x / 10)` as the last condition. We deducted two marks if you messed up the order.

Some students got the logical operator `&&` or `||` mixed up. We deducted two marks for each error.

END OF PAPER

This page is intentionally left blank.