NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING
FINAL ASSESSMENT FOR
Semester 1 AY2023/2024

CS1010 Programming Methodology

December 2023 Time Allowed 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This assessment paper contains 16 questions and comprises 10 printed pages, including this page.
2. Write or shade your answers in the answer sheet provided.
3. Please write and shade your student number in the corresponding box in the answer sheet.
4. The total marks for this assessment is 80. Answer **ALL** questions.
5. This is an **OPEN BOOK** assessment.
6. You can assume that in all the code given: (i) no overflow nor underflow will occur during execution; (ii) all given inputs are valid and fits within the specified variable type; (iii) compile without syntax errors; and (iv) all the required header files are already included.

Part I

Multiple Choice Questions (36 points)

- For each of the questions below, **shade your answer on the answer sheet.** Each question is worth 3 points.
- If multiple answers are equally appropriate, pick one and shade the chosen answer on the answer sheet. Do NOT shade more than one answer per question.
- If none of the answers is appropriate, shade `X` on the answer sheet.

1. (3 points) Consider a boolean function whose return value depends on three boolean variables, `x`, `y`, and `z`. The *only* three combinations that cause the function to return `true` are listed in the table below:

x	y	z
true	true	true
true	false	true
false	true	true

Which of the following `return` statement correctly implements the boolean function?

A. `return x || y || z;`

B. `return x && y && z;`

C. `return (x || y) && z;`

D. `return (x && y) || z;`

E. `return z;`

Shade `X` on the answer sheet if none of the choices above is correct.

2. (3 points) Consider the function below:

```
long ash(long m, long n) {
  long count = 0;
  while (m < n) {
    count++;
    m++;
    n--;
  }
  return count;
}
```

Suppose that `m < n`. Which of the expressions below best describes what the function computes?

A. `(n - m) / 2`

B. `(n - m) / 2 - 1`

C. `(n - m) / 2 + 1`

D. `(n - m + 1) / 2`

E. `(n - m - 1) / 2`

Shade `X` on the answer sheet if none of the choices above is correct.

3. (3 points) Consider the function below:

```
void eng(long m, long n) {
  do {
    m = do_something(m);
    n = do_something(n);
  } while (m != 0 && n > 0);

  if (m != 0 && n > 1) {
    m = 1; // Line W
  } else if (m == 0) {
    n = 1; // Line X
  } else if (n <= 0) {
    n = 0; // Line Y
  } else {
    m = n; // Line Z
  }
}
```

Which of the statements, labeled W, X, Y, and Z will never be executed for any values of `m` and `n`?

A. Line W

B. Line X

C. Line X and Line Y

D. Line Y and Line Z

E. Line W and Line Z

Shade X on the answer sheet if none of the choices above is correct.

4. (3 points) Consider the function `vee` below:

```
void vee(long h) {
  for (long i = 0; i < h; i += 1) {
    for (long space = 0; space <= i-1; space += 1) {
      putchar(' ');
    }
    putchar('#');
    for (long space = 0; space <= f(i); space += 1) {
      putchar(' ');
    }
    putchar('#');
    for (long space = 0; space <= i-1; space += 1) {
      putchar(' ');
    }
    putchar('\n');
  }
  putchar('\n');
}
```

The function will draw the following patterns, as examples.

When h is 3

```
#   #
 # #
  ##
```

When h is 4

```
#     #
 #   #
  # #
   ##
```

When h is 5

```
#       #
 #     #
  #   #
   # #
    ##
```

To achieve the pattern above, what should the function call `f(i, h)` return?

A. `(2 * h)`

B. `(2 * h) - i`

C. `(2 * h) - (2 * i)`

D. `(2 * h) - (2 * i) - 1`

E. `(2 * h) - (2 * i) - 3`

Shade X on the answer sheet if none of the choices above is correct.

5. (3 points) Consider the function below:

```
void bus(long *i, long *j) {
  // Find the element in the middle of *i and *j in the array.
  long mid = _________________;
}
```

The function is called with

```
long a[100];
bus(&a[0], &a[99]);
```

The variable `mid` should be initialized to the middle element in the array (i.e., `a[49]`). How should `mid` be initialized?

A. `*((i + j) / 2)`

B. `(*i + *j) / 2`

C. `*(i + j) / 2`

D. `*(i + (j - i) / 2)`

E. `*i + (*j - *i) / 2`

Shade X on the answer sheet if none of the choices above is correct.

6. (3 points) Consider the function below:

```
void doh(long n) {
  long *a;
  for (long i = 0; i < n; i += 1) {
    a = malloc(2 * sizeof(long));
    if (a == NULL) {
      cs1010_println_string("Not enough memory");
    }
    a[0] = 1;
  }
  free(a);
}
```

Which of the following statements about the function above is true? Assume that `n` > 1.

(i) There is a memory leaks if `malloc` is successful.

(ii) There is an illegal access to memory if `malloc` is successful.

(iii) There is an illegal access to memory if `malloc` is not successful.

A. (i) only

B. (iii) only

C. (i) and (ii) only

D. (i) and (iii) only

E. (ii) and (iii) only

Shade X on the answer sheet if none of the choices above is correct.

For Questions 7 and 8, consider the code snippet below, which allocates a 2D array with two rows and `n` columns. We assume that `calloc` does not fail and `n` is a positive number.

```
size_t n = cs1010_read_size_t();

long *canvas[2];
canvas[0] = calloc(2 * n, sizeof(long));
canvas[1] = canvas[0] + n;
canvas[0][n] = 1;  // Line U
free(canvas[0]);
free(canvas[1]);   // Line V
```

7. (3 points) Which of the following statements about Line U is correct?

   A. It sets `canvas[1][0]` to 1

   B. It crashes the program with a heap overflow error.

   C. It crashes the program with a stack overflow error.

   D. It crashes the program with a segmentation fault.

   E. It causes a memory leak.

   Shade X on the answer sheet if none of the choices above is correct.

8. (3 points) Which of the following statements about Line V is correct?

   A. It correctly ensures that there is no memory leak.

   B. It causes a memory leak.

   C. It crashes the program with a stack overflow error.

   D. It crashes the program since the memory pointed to by `canvas[1]` is not at the start of a memory region allocated with `calloc`.

   E. It crashes the program since the memory pointed to by `canvas[1]` is allocated on the stack.

   Shade X on the answer sheet if none of the choices above is correct.

9. (3 points) Suppose that the running time for the function `find` is $O(n)$, what is the running time of the function below, as a function of n?

```
for (long i = 1; i < n; i += 1) {
  if (n % (i * i) == i) {
    return find(n);
  }
}
```

   A. $O(\log n)$

   B. $O(\sqrt{n})$

   C. $O(n)$

   D. $O(n \log n)$

   E. $O(n^2)$

   Shade X on the answer sheet if none of the choices above is correct.

10. (3 points) The running time of a recursive algorithm can be characterized by the following recurrence relation:

$$T(n) = \begin{cases} T(n-1) + \log n & \text{if } n > 1 \\ 1, & \text{otherwise} \end{cases}$$

What is $T(n)$?

A. $O(\log n)$
B. $O(\log (n!))$
C. $O(n)$
D. $O(n \log n)$
E. $O(n^2)$

Shade X on the answer sheet if none of the choices above is correct.

11. (3 points) The final marks of the students in CS1010 are floating point numbers represented as `double` between 0 and 100. The marks were initially sorted in increasing order. Due to regrade requests from students, a small number of students have their marks adjusted and thus, the marks are no longer correctly sorted. The teaching team wants to re-sort the marks again in increasing order. Which of the following algorithms is the most appropriate for re-sorting the final marks after the regrade request?

A. Counting sort
B. Insertion sort
C. Bubble sort
D. Selection sort
E. Radix sort

Shade X on the answer sheet if none of the choices above is correct.

12. (3 points) Consider the implementation of the bubble sort and selection sort algorithms in class. Suppose we are given an array of size n that is inversely sorted, how many swaps are needed by each of the sorting algorithms?

	Bubble sort	Selection sort
A.	$O(n^2)$	$O(n^2)$
B.	$O(n^2)$	$O(n)$
C.	$O(n)$	$O(n^2)$
D.	$O(n)$	$O(n)$
E.	$O(1)$	$O(n)$

Shade X on the answer sheet if none of the choices above is correct.

Part II

Short Questions (44 points)

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

13. (6 points) Consider the function below, which calculates the number of digits in a positive integer n.

```
long ndigits(long n) {
  if (n < 10) {
    return 1;
  }
  return 1 + ndigits(n / 10);
}
```

Through formulating a recurrence relation, express the running time of the function using the Big-O notation as a function of n. Show your workings.

14. (10 points) Consider the program below.

```
void foo(long a[], long **p) {
  *p = a;
  (*p) += 1;
  // Line N
}

int main()
{
  long a[2] = {34, 56};
  long *ptr;
  foo(a, &ptr);
  cs1010_println_long(*ptr);
}
```

(a) (8 points) Draw the content of the call stack when the execution reaches Line N using the notations similar to what has been used in CS1010. Label all your stack frames, variables, and values on the call stack. You may use arrows to denote pointers, instead of using the actual memory address.

(b) (2 points) What would the program print to the standard output?

15. (22 points) Consider the following function `separate`, which takes an array of integer `a` with `n+1` (`n` > 0) elements as input.

```
void swap(long *a, long i, long j) {
  long temp = a[i];
  a[i] = a[j];
  a[j] = temp;
}

long separate(long *a, long n) {
  long i = -1;
  long j = n + 1;
  long x = a[0];
  while (true) {
    // Line B
    do {
      i += 1;
    } while (a[i] < x);
    // Line C
    do {
      j -= 1;
    } while (a[j] > x);
    // Line D
    if (i < j) {
      swap(a, i, j);
      // Line E
    } else {
      // Line F
      return j;
    }
  }
}
```

Denote `a[i..j]` to be the set { `a[i]`, `a[i+1]`, ... `a[j]` }. If `j` < `i` then the set is empty. We extend the comparison operations `<`, `>`, `<=`, and `>=` to set of elements. For example, we write `a[i..j] < x` if every element in the set `a[i..j]` is less than `x`. We write `a[i..j] <= a[k..m]` if every element in the set `a[i..j]` is less than or equal to every element in `a[k..m]`.

Using this notation, the goal of the function is to rearrange the array elements such as the assertion `a[0..j] <= a[j+1..n]` holds at Line F. We want to show that the function is correct.

(a) (4 points) First, trace through the function if it is invoked with:

```
long a[5] = { 8, 9, 6, 5, 7 };
separate(a, 4);
```

Write the values of variable `i`, `j` and the content of the array `a` every time the execution reaches Line E or Line F.

(b) (8 points) The invariant of the while loop is `a[0..i] <= x` and `a[j..n] >= x`. Assume that this assertion holds at Line B at the beginning of the while loop. By deriving the appropriate assertions at Line C and Line D, argue why the assertion is true at Line E.

(c) (4 points) Argue why at Line F, we can assert that `i == j || i - 1 == j`.

(d) (4 points) Use the invariant of the loop to argue why the assertion `a[0..j] <= a[j+1..n]` holds at Line F.

(e) (2 points) Express the running time of the function `separate` using big-O notation as a function of n. Explain your answer.

16. (6 points) The following function generates all possible binary strings (i.e., strings consisting of `'0'` and `'1'` only) of a given length `n`.

```
void generate(char str[], long n, long k) {
  if (k == n) {
    cs1010_println_string(str);
    return;
  }

  str[k] = '0';            // Line Q
  generate(str, n, k+1);   // Line R
  str[k] = '1';            // Line S
  generate(str, n, k+1);   // Line T
}
```

For example, when called with

```
char str[4] = { 0 };
generate(str, 3, 0);
```

It generates the following output:

```
000
001
010
011
100
101
110
111
```

The binary output is generated in increasing order of their decimal (base-10) values. In this question, we are interested in generating all possible binary strings such that two consecutive strings differ by at most one bit (one binary digit).

For example, we would like to generate the following when `n` is 2 and 3 respectively.

When `n` is 2

```
00
01
11
10
```

When `n` is 3

```
000
001
011
010
110
111
101
100
```

Assuming that the string `str` has been initialized to a string with `n` `'0'`s, replace the four statements labeled Q, R, S, and T in the function `generate` so that it generates the binary string in the desired order. Use only recursion without loops.

END OF PAPER