Family name ______Solution________

Given name(s) ____________________

Student number ____________________

York University
Faculty Science & Engineering / Faculty of Arts
Department of Computer Science & Engineering

Class Test 1

AK/AS/SC – CSE 3311 3.0
Software Design

2012 October 18

Instructions

1. This is a closed book examination. No examination aids are permitted – no calculators, telephones, etc.
2. All questions are to be attempted.
3. Answer each question in the space provided.
4. Each question is evaluated on the York University letter grade scale A+, A, ... , D, E, F.
5. Where descriptive answers are requested, use complete sentences and paragraphs
6. All programming and mathematical work is to be annotated; include comments explaining what you are doing.
7. Wherever appropriate, use diagrams to enhance your answers.
8. <u>If a question is ambiguous or unclear then please write your assumptions and proceed to answer the question</u>.
9. <u>You may write in pencil but class tests written in pencil are not re-evaluated</u>.
10. The test time is approximately 80 minutes.

Ques	Max	Mark
1	9	_____
2	9	_____
3	9	_____
4	9	_____
Total		_____
Letter grade		_____

Question 1

A How does the direct mapping rule apply to software design?

Describe the essence of slides 04-22, 14-11, 11-31..11-33. Textbook pages 47 & 54

B What does it mean for a function to have side effects?

I was looking for two ideas:

- "function that changes the state of an object has side effect"

- "an expression which contains a function with side effect may not be referentially transparent, because ..." (bonus)

C Why should software designers care about information hiding?

Summarize slides Designing classes part. Also look in textbook. Important to state that it makes it easier to maintain the program. Higher maintainability

Question 2

A What is a loop invariant?

Defines an invariant property of the variables used in a loop statement that is related to the work the loop does. It is not any assertion that is true throughout the loop execution.

B When is a loop invariant used?

Used at run time to verify the loop body is executing correctly. Used at design time to program a loop and/or to verify the correctness of a loop.

C Does the loop invariant depend on the postcondition (ensure)? You must justify your answer.

Yes! Since it should be used to prove the correctness of the loop. If you change the postcondition, then the loop invariant needs to be changed, and vice-versa, as there is a co-dependency.

D Does the loop invariant depend on the loop implementation? You must justify your answer.

Yes! This is an assertion about the loop property. It is about the variables in the loop and their relationships. If you change the implementation, then the loop invariant needs to be changed, and vice-versa, as there is a co-dependency.

Question 3

Using **mathematical notation**, complete the precondition, postcondition, loop invariant and loop variant for the following routine that moves the rightmost smallest element (the ≥ in the program text is important) within the array range A[lb .. ub] to A[ub]. Explain in English what you mean.

```
move_smallest(A: ARRAY[REAL], lb : INTEGER, ub : INTEGER) is
  require
```

argument_exists: $A \neq \text{void}$
valid_lower_upper: $A.\text{lower} \leq lb \wedge ub \leq A.\text{upper}$
need the clause as the parameters are not related to the array.

```
  local index, small_index : INTEGER
  from small_index := lb  ;  index := lb ;
  invariant
```

smallest_so_far: $\forall\, k : lb .. index \bullet A[small\_index] \leq A[k]$

rightmost: $\forall\, k : small\_index + 1 ..\ index \bullet A[small\_index] > A[k]$

```
  until index = ub    not upper, why lb and ub are parameters
  loop
    index := index + 1
    if A[small_index] >= A[index] then small_index := index end
variant
```

ub – index + 1

```
  end
  swap(A[small_index] , A[ub])
  ensure
```

smallest_in_position: $\forall\, k : 2 .. upper \bullet A[ub] \leq A[k]$

rightmost_moved: $\exists\, k : lb .. ub \bullet A'[ub] = A[k]$ -- swap was
$\wedge\ A'[k] = A[ub]$ -- done
$\wedge\ \forall\, j : k{+}1 .. ub \bullet A[lb] < A[k]$ -- rightmost moved
$\wedge\ \forall\, j : lb{+}1 .. ub \mid j \neq k \bullet A'[j] = A[j]$ -- no other changes

Followed your assumptions, if they made sense

```
end
```

Question 4

Using the given assertions, prove the following algorithm works correctly. The more mathematically precise you are and the better the annotation, the higher the grade because it is less ambiguous.

Pre-condition: a : INTEGER $\wedge$ $b \geq 0$
Post-condition: $z = abs(a + 2*b)$ – abs is the absolute value, $z > 0$
Loop invariant: $z = a + 2*(b - y)$

```
1 z := a
2 y := b
3 while y ≠ 0
4   z := z + 2
5   y := y - 1
6 end
7 if z < 0 then z := –z
```

Question 0: loop invariant is given
Question 1 – base case – Is loop invariant true after initialization phase, before loop body
Loop initialization occurs in lines 1 and 2 giving the relationships: $z = a \;\wedge\; y = b$
Substitute the values of z and y into the loop invariant to get the following expression.

$a = a + 2*(b - b)$ and simplifying gives $a = a$ which is true. The base case is established

Question 2 – inductive case: Assuming LI is true before executing loop body, and loop condition permits executing the body, is LI true after executing the loop body.
[Can work forwards or backwards; here we work forwards.]
Executing the body gives the relationships: $z' = z + 2 \;\wedge\; y' = y - 1$
Solve for z and u, and substitute into the loop invariant

$z = a + 2*(b - y)$ – the loop invariant
$\rightarrow z' - 2 = a + 2*(b - (y' + 1))$ – substitute for z and y
$\rightarrow z' = a + 2*(b - y' - 1) + 2$ – Add 2 to each side and do inner subtraction
$\rightarrow z' = a + 2*(b - y') - 2 + 2$ – Take the –1 out of the ()
$\rightarrow z' = a + 2*(b - y')$ – the 2's cancel

The final expression is the loop invariant at the end of the loop body. Since all we did was substitute equals for equals and rearrange terms using valid arithmetic operations, and the original LI is true, then it follows that the LI at the end of the loop body is true.

Question 3a: Does the loop terminate?
$y \geq 0$ at line 3 because $b \geq 0$ from the precondition and $y = b$ from line 2.
y is decremented by 1 every time around the loop, therefore eventually y must be equal to zero and the exit condition $y = 0$ must become true.

Question 3b: Does exit condition and LI imply post-condition?
At loop termination the loop invariant is true. The following substitutions and rearrangements are made

$z = a + 2*(b - y)$ -- the loop invariant
$\rightarrow z = a + 2*(b - 0)$ -- Substitute $y = 0$
$\rightarrow z = a + 2*b$ -- (1)

Now if at line 7 we have two cases.
Case 1: $z < 0$ and the then-phrase executes with the relationship $z' = -z$
As a consequence relationship (1) is equivalent to $z = abs(a + 2*b)$
Case 2: $z \geq 0$ and relationship (1) is equivalent to $z = abs(a + 2*b)$
In both case the relationship is $z = abs(a + 2*b)$ true. But this is exactly the post condition of the algorithm. As a consequence the algorithm is correct with respect to the given conditions.

For remarking you need to write a note stating clearly and exactly where you believe your grade should be increased or decreased. Remember that the grade is a qualitative one. You need to explain, if you think your grade should go up, why you believe the quality of your answer is good (B) and not competent (C+), or, if you think the grade should go down, why you believe the quality of your answer is very good (B+) and not excellent (A).

The entire test will be reevaluated. Your grade may go up, it may stay the same, or it may go down. I will look over the entire test and see if the grades good, excellent, minimal, etc are applicable to the work as a whole (see the web page on grading in the course) independent of the points assigned to the parts.