Student ID: .....

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VUW VICTORIA UNIVERSITY OF WELLINGTON

EXAMINATIONS – 2014

MID-TERM

SWEN 224
Formal Foundations of Programming

Time Allowed: 50 minutes

Instructions:
- There are 50 possible marks.
- You may Answer all six questions but **your final mark will be the sum of the marks from only your best five answers.**
- Write your answers in the boxes provided.
- If additional space is required you may use one of the spare pages, and indicate where your answer is in the box for that question.
- Write clearly and cross out rough working and anything else you don't want marked.
- Non-electronic foreign language dictionaries are allowed.
- Calculators ARE NOT ALLOWED (and not required).
- No other reference material is allowed.

Question	Topic	Marks	Achieved
1.	Understanding Assertions A	10	
2.	Writing Assertions	10	
3.	Verification and Preconditions	10	
4.	Verification and Preconditions	10	
5.	Loops	10	
6.	Whiley	10	
	Total	50	

Question 1. Understanding Assertions - A [10 marks]

Consider the following assertion:

$$\exists i \bullet 0 \leq i \leq (|A| - 1) \wedge (\exists j \bullet 0 \leq j \leq (|A| - 1) \wedge (2 * A[j]) = A[i])$$

where A is assumed to be an array of numbers, with indexes starting from 0 and $|A|$ is the number of elements in the array.

(a) [3 marks] State **in you own words** what the assertion means. **Do not** translate one symbol at a time as this is unintelligible.

(b) [2 marks] For each of the following arrays, say whether the assertion is true or false for that array:

(i) $A = (2, 3, 5)$

(ii) $A = (5, 3, 15, 6)$

Consider the following assertion:

$$\forall j \bullet 0 \leq j \leq (|A| - 1) \rightarrow (\exists i \bullet 0 \leq i \leq (|A| - 1) \wedge (i \neq j \wedge A[i] = A[j] \vee (A[j] + 5) \leq A[i]))$$

where A is assumed to be an array of numbers, with indexes starting from 0.

(c) [2 marks] State **in you own words** what the assertion means. **Do not** translate one symbol at a time as this is unintelligible.

(d) [3 marks] For each of the following arrays, say whether the assertion is true or false for that array:

(i) $A = (1, 2, 5)$

(ii) $A = (3, 3, 13)$

(iii) $A = (11, 15, 15)$

Question 2. Writing Assertions [10 marks]

Write an assertion (a mathematical/logical expression) to formalise each of the following statements.

(a) [1 mark]

Integer i is less than x and equal to or greater then y.

(b) [1 mark]

All values in the array A (size of A is $|A|$) are greater than their index.

(c) [1 mark]

No value in the array A is either 1 or 2.

(d) [2 marks] The values in array A are all different and in ascending order.

(e) [2 marks] All value in array A occurs twice.

(f) [3 marks] All values in array A occurs only twice.

Question 3. Computing precondition [10 marks]

For each of the following programs calculate the conditions by working from the given post condition Q and working backwards applying Hoare logic.

All variables, $x, y, z, w \ldots$ are integers - $\mathbb{Z}$ or, A, arrays of integers.

(a) [3 marks]

- P=
- $y := 3x;$
-
- $y := y + 2x;$
-
- $x := y + 2;$
- $\{x + y > 20\}$

(b) [4 marks]

P =

P1

S1 $x := x - 5;$

Q

P2

S2 $x := 20;$

Qa

S2 $x := 2x + 1;$

Q

S **If** $(x > 20)$ **then** $x := x - 5$**; else** $x := 2x + 1; x := 20$**; fi**

Q $\{30 \leq x \leq 40\}$

(c) [3 marks]

P =

P1

Sb $y := x + 1;$

Sa $x := x - y;$

Q

P2

S2

Q

S **If** $(x < 2)$ **then** $y := x + 1; x := x - y;$ **fi**

Q $\{x < 1\}$

Question 4. Computing precondition [10 marks]

(a) [5 marks]

- P=
- $w := 2;$
-
- $x := 3;$
-
- $A[x] := 4;$
-
- $z := A[w];$
- $\{z = 4\}$

(b) [5 marks]

P =

P1

Sb $x := x + 1;$

Sa $A[x] := y;$

Q

P2

Sb $A[x] := y;$

Sa $x := x + 1;$

Q

S **If** $(x > 20)$ **then** $x := x + 1; A[x] := y;$ **else** $A[x] := y; x := x + 1;$**fi**

Q $\{A[x] = 1\}$

Question 5. Loops [10 marks]

(a) [5 marks] Define the three loop verification conditions for:

while B then {**S**}

Use: loop invariant, **L**; while guard, **B**; body of the while , **S**; condition before the loop, **P** and condition after the loop, **Q**.

The following code should compute $2y$ and stores the result in x.

Pre {???}

```
i:=0;
x:=0;
```

P {???}

```
while (i< y) then
  {i:=i+1;
   x:=2+x;}
```

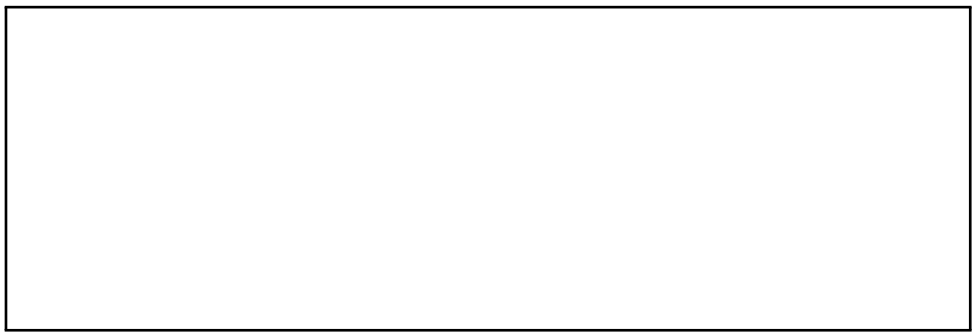
Q $\{x = 2y\}$

Loop invariant **L** =

Loop precondition **P** =

Code precondition **Pre** =

1. write down the three loop verification conditions for this loop
2. are all three conditions true



(i) [5 marks] For the program below compute **L**, **B** and **Q**. You may make use of the ghost variables or you may choose not to.

loop invariant **L**

guard **B**

what is decreasing

Show that $\mathbf{L} \wedge \neg \mathbf{B} \rightarrow \mathbf{Q}$

Fill in all the conditions. You may write **B** and **L** as short hand for the definitions you wrote above or you can write them out.

- **Pre**
- $i := 0;$
-
- $x := 1;$
-
- $xg := x; ig := i;$
- **P**
- **while** $(i < y)$ **then** $\{$
-
- $i := i + 1;$
- $x := 2x$
-
- $\}$
- $\{x > 8\}$

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 6. Whiley [10 marks]

In the two subquestions that follow you can make use of a function **setequal**.

The function **setequal** takes two integer lists and returns true if and only if the lists contain the same set of elements see below:

```
function setequal([int] x, [int] y) => (bool r)
        ensures r <==> (all{v in x | some{w in y | w ==v}} &&
                        all{v in y | some{w in x | w ==v}}):
```

The code of **setequal** is included for reference only. If you understand the specification **you do not need to read the implementation**.

```
function setequal([int] x, [int] y) => (bool r)
        ensures r <==> (all{v in x | some{w in y | w ==v}} &&
                        all{v in y | some{w in x | w ==v}}):
    bool b = true
    for v in y:
        if !isin(x, v):
            return false
    return b

function isin([int] x, int t) => (bool r):
     bool b = false
     for v in x:
        if v == t:
           return true
     return b
```

(a) [5 marks] Read Whiley **function pivot**. This function has two parameters, **row** is an array of integers and **p** an integer. it returns a record of **type split** containing two integer arrays **small** and **large**. The array **small** contains all element in **row** that are smaller than **p** and array **large** contains all the elements that are not smaller.

```
type    split is {[int] small, [int] large}
function pivot([int] row, int p) => (split  r)
  ensures ????:
    [int] l = []
    [int] s = []
    for i in 0..|row|
        where ????:
        if row[i] < p:
            s = s ++[row[i]]
        else:
            l = l ++[row[i]]
    split sp =     {small: s, large: l }
    return sp
```

Define the post condition for **pivot**.

Define the loop invariants for **pivot**.

(b) [5 marks] The **function merge** takes two sorted lists and returns the list formed by merging their elements.

```
function merge([int] x, [int] y) => ([int] o)
    requires ????
    ensures  ????:
    int i = 0
    int j = 0
    [int] arr = []
    while i < |x|:
        while j < |y| && x[i] > y[j]
           where ??1??:
            arr = arr ++ [y[j]]
            j=j+1
        arr = arr ++ [x[i]]
        i=i+1
    while j < |y|
       where ??2??:
        arr = arr ++ [y[j]]
        j=j+1

    return arr
```

Define the pre condition and post condition for **merge**.

Define the two loop invariants marked **where ??1??** and **where ??2??** .

****