# Midterm. Engi 9869

Theodore Norvell ©

2008

Total marks: **35**
**Name:**
**Student #:**
**Q0 [10]**
**Complete** the following proof outline validly. Keep it as simple as possible.

---

## $a = b = false$

# Global Inv: ______________________________
**co**

## ______________________________
$a, w := true, 13$

## ______________________________
$\langle \mathbf{await}(b) \rangle$

## ______________________________
$x := f(w, y)$

## ______________________________
//

## ______________________________
$b, y := true, 42$

## ______________________________
$\langle \mathbf{await}(a) \rangle$

## ______________________________
$z := g(w, y)$

## ______________________________
**oc**

## $x = f(13, 42) \wedge z = g(13, 42)$

---

Name: ______________________ Student #: ________________

**Q1 [10] Future.** A "future" is an object that either represents a value, or will represent a value in the future. Typically a future is returned from a function that may spin off a new thread to compute the value. For example

---

```
Future future = someObj.someMethod() ;
...
Object value = future.get() ;
```

---

The thread getting the value may have to wait until the value has been put. Using Java notation **design and implement** a Future class. It should support the following methods.

---

```
public void put( Object pValue ) ;
public Object get() ;
```

---

You may use my Signal-and-Wait monitor package, if you want.

Name: ______________________ Student #: ________________

**Q2 [5]. Remote Procedures.**
In what ways do remote procedure calls differ from ordinary (local) procedure calls?

---

**Q3 [10] Bag of tasks product.** Consider the problem of computing the 'product' of all items in an array $a[0..N-1]$.

$$a[0] \otimes a[1] \otimes ... \otimes a[N-1]$$

We can assume that $\otimes$ is some associative operator, which may be time consuming to compute.

For simplicity assume $N = 2^n$ where $n$ is an integer.

We can make use of a tree structure, filling in an array $A[0..n, 0..N-1]$.So that $A[0,j] = a[j]$ and $A[i,j] = A[i, 2 \times j] \otimes A[i, 2 \times j + 1]$, where $0 < i \leq n$ and $0 \leq j \leq 2^{n-i}$. This is illustrated for $N = 8$, $n = 3$:

| A | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | $a[0]$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | $a[7]$ |
| 1 | $A[0,0] \otimes A[0,1]$ | $A[0,2] \otimes A[0,3]$ | $A[0,4] \otimes A[0,5]$ | $A[0,6] \otimes A[0,7]$ | | | | |
| 2 | $A[1,0] \otimes A[1,1]$ | $A[1,2] \otimes A[1,3]$ | | | | | | |
| 3 | $A[2,0] \otimes A[2,1]$ | | | | | | | |

**Design** a bag of tasks algorithm in which each task consists of filling in one element of $A$.

In case it helps, let's define the following functions to help navigate the tree

$$\begin{aligned}
\textit{sibling}((i,j)) &= (i, j+1), \text{ if } j \text{ is even and } i < n \\
\textit{sibling}((i,j)) &= (i, j-1), \text{ if } j \text{ is odd and } i < n \\
\textit{parent}((i,j)) &= (i+1, \lfloor j \div 2 \rfloor), \text{ if } i < n \\
\textit{chidren}((i,j)) &= \{(i-1, 2j), (i-1, 2j+1)\}, \text{ if } i > 0
\end{aligned}$$

Use the course's design notation. Feel free to declare any auxiliary data structures you would like.

Hint: Ensure that each 'parent' task goes in the bag only after both of its child tasks have filled in their elements of the $A$ array.