Name: ______________________________

16.unified
Introduction to Computers and Programming

Examination II- Solutions

5/19/04
9-10am

Prof. I. Kristina Lundqvist
Spring 2004

Question 1 (30)	
Question 2 (5)	
Question 3 (10)	
Question 4 (10)	
Question 5 (10)	
Question 6 (10)	
Question 7 (15)	
Question 8 (10)	
Total 100	

You have 55 minutes to take this examination. Do not begin until you are instructed to do so. This is a closed book examination. No external materials are permitted, including calculators or other electronic devices. All answers must be written in the examination paper. This examination consists of 8 questions and 12 pages (not including this cover page). Count the number of pages in the examination paper before beginning and immediately report any discrepancy to the invigilator. Should you need to do so, you may continue your answers on the back of pages.

Do not forget to write your name on each page.

Name: ______________________________

Problem 1 – A generic queue (30 points)

Given the following Generic_Queue Specification and Body,

```
-- specification for generic queue implementation
generic
   Size : Positive;
   type Item is private;
   package Queue is

   procedure Enqueue(
         E : in     Item );
   procedure Dequeue(
         E :    out Item );
   Overflow, Underflow : exception;
end Queue;
```

```
-- package body for generic queue implementation

package body Queue is
   type Table is array (Positive range <>) of Integer;
   Space : Table (1 .. Size);
   Head: Natural           := 1;
   Tail: Natural:= 1;
   procedure Enqueue(
         E : in     Item ) is
   begin
      if Tail>= Size then
         raise Overflow;
      end if;
      Space(Tail) := E;
      Tail:= Tail+ 1;
   end Enqueue;

   procedure Dequeue(
         E :    out Item ) is
   begin
      if Head = 0 then
         raise Underflow;
      end if;
      E := Space(Head);
      Head:= Head - 1;
      if Head /= 0 then
         for I in Head+1 .. Tail-2 loop
            Space(I):= Space(I+1);
         end loop;
         Tail:= Tail -1;
      end if;

   end Dequeue;

end Queue;
```

Part a. Will the generic package compile? Justify your answer. (5 points)

Given that there are no syntax errors in the code listing shown above.

The package will **not** compile, because the queue is defined as an array of integers, while enqueue and dequeue are both trying to add a generic element.

Name: ________________________________

Part b. Can you create two instances of the package i.e. a *character queues* and *integer queues* from the code shown above. (Answer with one word: ***yes*** or ***no***) (3 points)

No

Part c. (6 points)

(i) If ***yes*** in part b, Define the Ada95 instantiation for a character queue and an integer queue.

This gets six points even if Parts a, b were wrong.

Character Queue

```
package Queue_Char is new Queue(
      Size => 5,
      Item => Character);
```

Integer Queue

```
package Queue_Int is new Queue(
      Size => 5,
      Item => Integer);
```

(ii) If ***no*** in part b, Which line(s) in the specification or body do you have to change? List the line number and modification to be made.

```
4.    type Table is array (Positive range <>) of Integer;
```

Has to be changed to:

```
4.    type Table is array (Positive range <>) of Item;
```


Name: ________________________________

Part d. (16 points)

Modify the Enqueue procedure for a **circular** queue. Include exception handlers to handle overflow.

Hint: How do you check if the queue is full or empty?

```
procedure Enqueue (
      Element : in     Item ) is

begin
   if Tail+1 = Head then
      raise Overflow;
   end if;
   Space(Tail) := Element;
   Tail:= Tail+ 1;-- moves tail to the next location
   if Tail = Size then -- make the tail
                       -- point to first location
      Tail :=1;
   end if;

exception
   when Overflow =>
      Put_Line("Circular Queue is full");
end Enqueue;
```

Name: ______________________________

Problem 2 – Induction Proof (5 points)

Prove using induction that the sum of the first **n** positive integers is **n*(n+1)/2**.

1 + 2 + 3 + .. + n = n*(n+1)/2

Base Case

When n = 1,

n*(n+1)/2 = 1*(2)/2
= 1

Induction Hypothesis: Let the theorem be true for j= n.

Inductive Step:

When j = n+1, then

Sum	=	1 + 2 + 3 + .. + n + (n+1)	
	=	n*(n+1)/2 + (n+1)	[By Induction Hypothesis]
	=	$((n^2+n)+ (2n+2))/2$	[Arithmetic]
	=	$(n^2+3n+2)/2$	[Arithmetic]
	=	(n+1)*(n+2)/2	[Arithmetic]

By theorem
Sum = (n+1)*(n+2)/2

Hence

For all n ≥ 1, the sum of the first **n** positive integers is **n*(n+1)/2**.

Name: ______________________________

Problem 3 – Variant record (10 points)

Define a variant record to holds aircraft information for two different types of aircraft: Fighters and Bombers

Both kinds of Aircraft have three common fields:
- **ID** is of type **integer**
- **Call_Sign** is a **string** of maximum 20 characters
- **Aircraft_Type** is an **enumerated** type with values ‘fighter’ and ‘bomber’

In addition to the three common fields, Fighter Aircraft have fields:

- **Aircraft_Mach** is a **record** with fields
  - **Top_Mach** of type **float**
  - **Cruise_Mach** of type **float**
- **Max_Range** is of type **integer**

In addition to the three common fields, Bomber Aircraft have fields:
- **Crew_Size** of type **positive**
- **Payload** of type **integer**

```
type Aircraft_Type is(Fighter,Bomber);
type Fighter_Mach is record
     Top_Mach : Float;
     Cruise_Mach : Float;
end record;

type Aircraft_Record(Kind: Aircraft_Type) is
record
   ID : Integer;
   Call_Sign: String(1..20);
   Type : Aircraft_Type;

     case Kind is
      when Fighter =>
         Aircraft_Mach : Fighter_Mach;
          Max_Range : integer;

      when Bomber=>
         Crew_Size: Positive;
          Payload : integer;
   end case;
end record;
```

Name: ________________________________

Problem 4 – Tree traversal (10 points)

Part a. Define recursive algorithms to traverse a tree in *preorder* and *postorder* (6 points)
Fill in the preconditions (constraints/input parameters), postconditions(result/ouput parameters)

Traverse Preorder

Preconditions: root node of tree
Postconditions: displayed the tree by traversing preorder
Constraints : the tree is assumed to be well formed (no dangling references).

Pseudocode:

1. If root = null then exit program
2. display root.element.
3. traverse the left subtree using root.Left_Child as the root.
4. traverse the right subtree, using root.Right_Child as the root.

Traverse Postorder

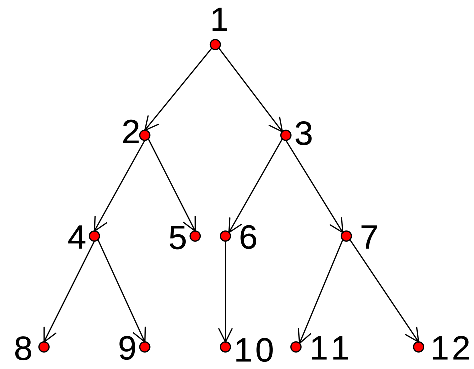
Preconditions: root node of tree
Postconditions: displayed the tree by traversing in postorder
Constraints : the tree is assumed to be well formed (no dangling references).

Pseudocode:

1. If root = null then exit program
2. traverse the left subtree using root.Left_Child as the root.
3. traverse the right subtree, using root.Right_Child as the root.
4. display root.element.

Name: ________________________________

Part b. For the given tree shown below: What is the ***inorder*** traversal of the tree.
(4 points)



8,4,9,2,5,1,10,6,3,11,7,12

Name: ________________________________

Problem 5 – Proof (10 points)

Part a. What is a fully connected graph? (3 points)

A fully connected graph is a graph in which every node is connected to every other node. A fully connected graph with *n* nodes is denoted by K_n.

Part b. (7 Points)

Prove that the sum of the degrees of all nodes in a fully connected graph of n nodes is **n*(n-1).**

Base Case
When n = 1, the graph has one node and no edges, so the degree is zero.

n*(n-1)	=	1*0
	=	0

Induction Hypothesis: Let the theorem be true for a graph with n nodes.

Inductive Step:

When the graph has n+1 nodes, the $(n+1)^{th}$ node has to be connected to the remaining n nodes of the fully connected graph, and hence has degree n. The n nodes of the Kn graph have their degree increased by 1. Hence the total increase to the sum of the degrees is 2*n.

=	n*(n-1) + 2*n	[By Induction Hypothesis]
=	n^2+n	[Arithmetic]
=	(n+1)*n	[Arithmetic]

By theorem
Sum of all the degrees of the nodes of a fully connected graph with n+1 nodes is
n*(n+1)
Hence

∀n ≥ 1, the sum of the degrees of a fully connected graph with **n** nodes= **n*(n-1)**

Name: ________________________________

Problem 6 – Logic (10 points)

Part a. Formally prove **T → ¬(P ∨ R)**, given the following hypothesis 1 and 2: (2 points)

1. P ∨ R → S
2. T → ¬S
3. ¬S → ¬(P ∨ R) [contrapositive of 1]
4. **T → ¬(P ∨ R)** [2,3 Transitivity of ->]

Part b. Translate the following four sentences of English into the language of predicate logic. **E(x)** represents x is even, and **O(x)** represents x is odd. (8 points)

1. 2 is even

   Even(2)

2. Not every integer is even

   ∃x ¬Even(x)

3. Some integers are even and some are odd

   ∃x∃y (Even(x) and ¬Even(y))

4. If an integer is not even, then it is odd

   ¬Even(x) → Odd(x)

Name: ______________________________

Problem 7 **(15 points)**

Part a. Convert the following POS (Product of Sum) expression into SOP (Sum of Product) form. (5 points)

Hint: Think about negation.

$$(A+B+C)\bullet(\overline{A}+\overline{B}+\overline{C})\bullet(A+\overline{B}+\overline{C})\bullet(A+\overline{B}+C)$$

Negating the expression above,

$$(\overline{A}\bullet\overline{B}\bullet\overline{C})+(A\bullet B\bullet C)+(\overline{A}\bullet B\bullet C)+(\overline{A}\bullet B\bullet\overline{C})$$

These are the 0's, from the crib sheet, the 1's can be found.

$$(\overline{A}\bullet\overline{B}\bullet C)+(A\bullet\overline{B}\bullet\overline{C})+(A\bullet\overline{B}\bullet C)+(A\bullet B\bullet\overline{C})$$

Part b. Simplify the SOP expression derived above using K-Maps (5 points)

	$\overline{B}\bullet\overline{C}$	$\overline{B}\bullet C$	$B\bullet C$	$B\bullet\overline{C}$
$\overline{A}$	0	1	0	0
A	1	1	0	1

Simplified Expression is $(\overline{B}\bullet C)+(A\bullet\overline{C})$

Part c. What are the minterms that go into a four-variable K-Map (5 points)

	$\overline{B}\bullet\overline{C}$	$\overline{B}\bullet C$	$B\bullet C$	$B\bullet\overline{C}$
$\overline{A}\bullet\overline{D}$				
$\overline{A}\bullet D$				
$A\bullet D$				
$A\bullet\overline{D}$				

Name: ______________________________

Problem 8 – Multiple choice **(10 points)**

Multiple Choice Questions. For each question, select the correct answer from the choices, and **write the chosen letter in the box provided** next to each question.

Answer

1. Prof. Dewar mentioned during his guest lecture that "Visual Basic" still is one of the two most commonly used programming languages, which of the following languages is the other of the top 2 most used programming languages in the world?

   a. Java
   b. C
   c. Cobol
   d. Ada

C

2. Heidi Perry explained four commonly used Software Life Cycles, which four?

   a. The Waterfall, Incremental, Acquisition, and Spiral models
   b. The Niagara, Incremental, Evolutionary, and Spiral models
   c. The Waterfall, Incremental, Evolutionary, and Spiral models
   d. The Niagara, Incremental, Revolutionary, and Spiral models

C

3. What is "*cyclomatic complexity*"?

   a. Number of independent paths needed to execute all statements and conditions in a program at least once
   b. A theoretical measure used in complexity theory, that describes the asymptotic ***lower*** bound of a function in terms of another, usually simpler, function
   c. A measure of the number of times a recursive function will recursively call itself/be executed.

A

Name: ________________________________

4. Coupling is a property of a collection of software modules. Which out of the following versions of coupling has the **highest/worst** coupling?

B

a. **Stamp** coupling: Two modules are stamp coupled if they communicate through a composite data structure
b. **Content** coupling: Two modules are said to be content coupled when they share code.
c. **Data** coupling: Two modules are data coupled if they communicate via a parameter
d. **Common** coupling: Two modules are said to be common coupled when both reference the same shared/global data

5. I would like to have 2 free points on this quiz

☺

a. Yes
b. Yes please
c. Yes, pretty please
d. No, I believe that **Nothing** is for free in life!