<u>16.unified</u>

Introduction to Computer Programming

Examination

11/18/05

Professor I. Kristina Lundqvist,
Lecturers Heidi Perry and Gustaf Naeser

Fall 2005

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 should consist of **11** pages (including this cover sheet). Count the number of pages and immediately report any discrepancy. Should you need to do so, you may continue your answer on the back pages. Put your name at the bottom of each page of this exam.

Question 1 True/False (10)	
Question 2 Read Ada Code (20)	
Question 3: Recursion (10)	
Question 4: Arrays (15)	
Question 5: Number Conversions (15)	
Question 6: Computer Architecture (10)	
Question 7: Rover (20)	

 Name_______________________________

Problem 1

True or False (10p)

(1p each)

For each statement, indicate if it is true (T) or false (F).

The strong typing of Ada95 prevents users from confusing variables of different types.	T
Iteration and recursion solves the same kind of problem: repeated execution of statements.	T
A string in Ada95 can be treated as records of characters.	F
Variable declarations implicitly set default values of the declared variable.	F
Variables used as actual parameters must have the same name as the name of the subprogram's formal parameters.	F
All kinds of loops can be described using for loops.	F
A function that returns two values must be implemented as a procedure.	T
Distinct datatypes, e.g. type My_Type is new Integer, are for creating types that should not be confused with other types using the same symbols (possible values).	T
Subtypes are assignment compatible with the base type.	T
You can store both positive and negative numbers using 2-complement	T

Comments

1 Among many things
2 Recursion loops code according to the programmer's design
3 A string is an array of characters
4 Variables must be given values (the declaration just allocates the memory)
5 No, the formal parameters are separate from the actual ones
6 The general loop statement is the "loop"
7 Functions can only return a single value whereas procedures can have several out parameters.
8 True
9 Subtypes are (Subtype_Var := Basetype_Var_or_Value is allowed)
10 True

Problem 2 *Reading Ada Code (20p)*

Our programmer friend Joe D has written the program `main.adb`. Predict what the output of Joe's program will be at commented lines A through F.

`main.adb`

```
with Ada.Integer_Text_Io;
use Ada.Integer_Text_Io;

procedure Main is

   A : Integer := 2;
   B : Integer := 5;
   C : Integer := 4;

   function F1 (X: Integer;Y :Integer) return Integer is
      C : Integer; --this C is local to the function and will not change the global C
   begin
      C := X+Y;
      return C;
   end;

   procedure F2 (X: in out Integer; Y : in Integer) is --X is in out and can be changed!
   begin
      A := X; --the global A is changed!
      X := A*Y; --the variable coming in through X is changed!
   end;

begin
   B:=A*B; --B is assigned A * 5 = 2 * 5 = 10
   A:=F1(B,C); --A is assigned F1(10, 4) = (10 + 4) = 14
   Put(A);    -- Part A

   B:=B+C; --B is assigned B + C = 10 + 4 = 14
   Put(B);    -- Part B

   A:=A-C; --A is assigned A – C = 14 – 4 = 10
   F2(B,C); --A is set to B = 14, B (formal param X) is set to 14 * 4 = 56
   Put(B);    -- Part C
   Put(A);    -- Part D

   C:=F1(A,B); --C is assigned (A + B) = 14 + 56 = 70
   Put(C );    -- Part E

   for C in 1..A loop --This C is local to the loop and will be 1..14
      B:=B+1; -- for each lap increase B => 14 laps => increase B by 14 = 56 + 14 = 70
   end loop;
   Put(B);    -- Part F
   Put(C);    -- Part G

end Main;
```

Part A = 14	Part D = 14	Part G = 70
Part B = 14	Part E = 70	
Part C = 56	Part F = 70	

 Name_______________________________

Problem 3 *Recursion (10p)*

What is wrong with the following recursive function and how should it be corrected? (Hint, look at the results from the two calls to it.)

Indicate the change that should be done and give a short motivation/explanation of no more than two lines.

recursiveaddition.adb

```
with Ada.Integer_Text_IO;
use Ada.Integer_Text_IO;

procedure RecursiveAddition is

   function Add_Every_Other_From_Zero(N : Integer) return Integer is
   begin
      if (N = 0) then
         return 0;
      else
         return N + Add_Every_Other_From_Zero(N-2);
      end if;
   end Add_Every_Other_From_Zero;

begin
   Put(Add_Every_Other_From_Zero(8));
   Put(Add_Every_Other_From_Zero(7));
end RecursiveAddition;
```

The problem with the function is that it doesn't handle odd numbers. (Calling it with 8 is ok, it will take away 2 until it reaches the base case, 0, where the recursion will terminate, but calling it with an odd number, e.g., 7 will take away 2 until the base case is reached… but the case will never occur! Starting with an odd numbers will eventually reach 1 from which 2 will be taken away ending up with -1, from which 2 will be taken away from which 2 will be taken away… Infinite recursion!)

There are several possible solutions.

1 Fix the base case, e.g.,
"if (N <= 0) then" or "if (N = 0 or N = -1) then" or similar

2 Make the initial number odd (kind of solves the problem…)
"return N + Add_Every_Other_From_Zero(N-2);" is changed to
"if (N mod 2 = 0) then
    return N + Add_Every_Other_From_Zero(N-2);
 else
    return N + Add_Every_Other_From_Zero(N-1);
 end if;"
or similar

 Name_______________________________

Problem 4 *Arrays and Records (15p)*

a. Consider the program `arrayinit.adb` .

arrayinit.adb

```
with Ada.Integer_Text_IO;
use Ada.Integer_Text_IO;

procedure ArrayInit is
   type TenArray is array (1..10) of Integer;
   Chunk : TenArray;
begin
   for I in TenArray'Range loop
      Put(Chunk(I));
   end loop;
end ArrayInit;
```

What warnings or errors can be expected during compilation of the program? (3p)
arrayinit.adb:6:04: warning: “Chunk” is never assigned a value or anything indicating that there is a problem with the use of a variable which has not been initialized.

What will the output be when running the program? (2p)
The memory where the array is allocated contains “random” values when the array is allocated (declared) and these values will be printed, i.e., garbage.

Write code to correct the problem and indicate where the code should go. (5p)
Anything that initializes the contents of the array prior to the “Put” is ok.
Examples :

At the declaration of the “Chunk” variable:

“Chunk : TenArray := (0,0,0,0,0,0,0,0,0,0);” or
“Chunk : TenArray := (others => 0)” or similar

Before the loop:

```
“for I in TenArray’Range loop
     Chunk(I) := 0;
 end loop;”
```

Inside the loop, making the loop:

```
“for I in TenArray’Range loop
     Chunk(I) := 0;
     Put(Chunk(I));
 end loop;”
```

or anything similar.

Any value can be used (instead of the “0”).

b. Consider the program in `nutsandbolts.adb`. (5 p)

nutsandbolts.adb

```
with Ada.Text_Io, Ada.Integer_Text_Io;
use Ada.Text_Io, Ada.Integer_Text_Io;

procedure NutsAndBolts is

   type Description is
      record
         L    : Integer;
         H    : Integer;
         W    : Integer;
         Cost : Float;
      end record;

   type Sets is array (1..30) of Description;
   type Part is (Bolt, Nut, Cap);
   type Catalog is array (Part'range) of Sets;

   type Warehouse is
   record
      Bolts_R_Us: Catalog;
      Hold_Tighter : Catalog;
   end record;

   Dist_Center : Warehouse;

begin
   -- Set length (L) of the 10th Nut in the Bolts_R_Us Catalog to 10 HERE.


end NutsAndBolts;
```

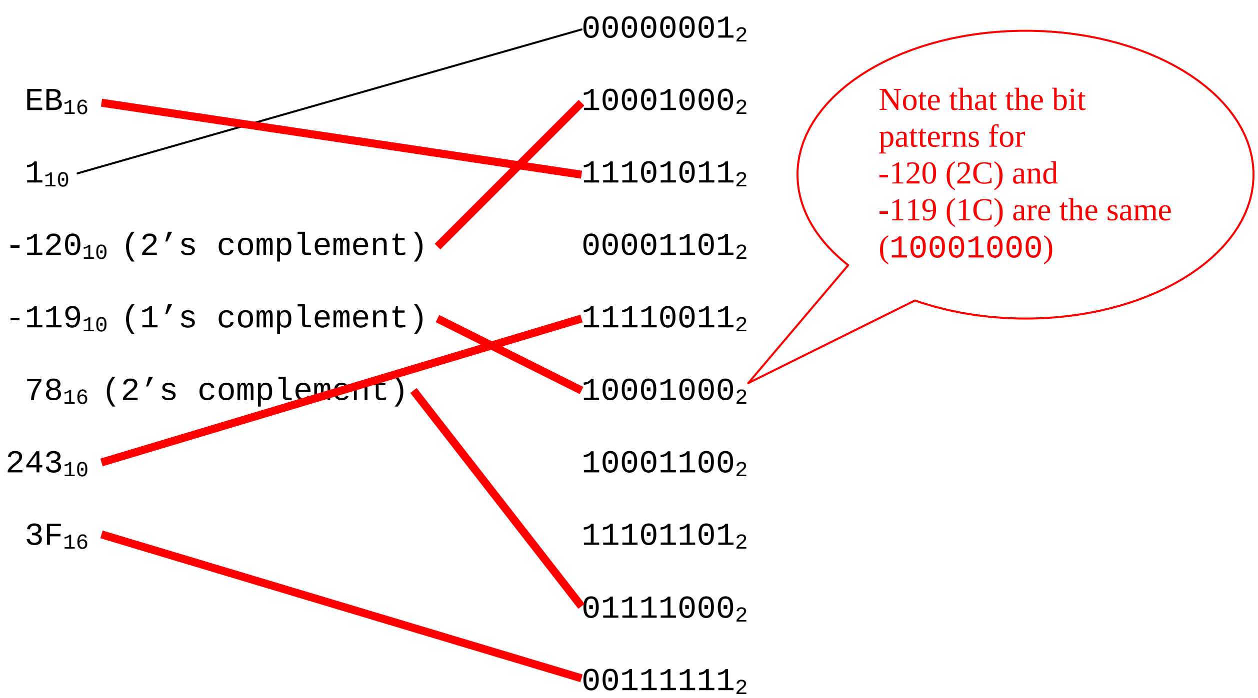
How would you go about setting the length (L) of the 10th Nut in the Bolts_R_Us Catalog to 10? Write the statement for the assignment in the code above. (10p)

Dist_Center.Bolts_R_Us(Nut)(10).L := 10;

 Name________________________________

Problem 5 *Number Conversions (15 P)*

a. Match the decimal and hexadecimal numeric values on the left to their equivalent binary numeric values on the right. (12p)



b. Convert the following floating point number to decimal notation. Note: The number has been stored using scientific notation, i.e., the leftmost 1 has been removed. (3p)

111011001100 with 1 signbit, 4 bits exponent (excess4) and 7 bits mantissa =

1	1101	11001100
sign	exponent	mantissa

The sign bit is set so the number is negative.

Yes, excess4 allows for large numbers when there are 16 possible values (4 bits). To get the number of steps the float point should be moved 4 should be removed from the "1101" => "1001" = 9 steps.

The one removed from the mantissa ""(when stored using scientific notation) should be reinserted => 1.11001100, move float point nine positions => 1110011000 = 920
So,

 Name________________________________

Problem 6 ***Computer Architecture (10 P)***

Assume you have an 8-bit computer that contains two registers (R1 and R2) to contain data the CPU will use for arithmetic operations. This computer is designed to process the following legal Opcodes:

000 = stop
001 = load data stored at indicated address into R1
010 = load data stored at indicated address into R2
011 = store R1 into address location given
100 = store R2 into address location given
101 = add R1 to R2 (results stored in R2)

The three most significant bits contain the op-code. The remaining 5 bits contain the address of any data that is required by the operation.

a. The machine level program that is written below is equivalent to the Ada statement of Y:=X + Y + Z; where X, Y, and Z are stored in addresses 10_{10}, 11_{10} and 12_{10} respectively. Fill in the meaning of each of the statements. The first one has already been done for you.

OpCode	Address	Meaning
010	01011 (11_{10})	Load data stored at address 11_{10} (the # Y) into R2 (=00001001)
001	01010 (10_{10})	Load data stored at address 10_{10} (the # X) into R1 (=00000011)
101	00000	Add R1 and R2, store in R2 (=00001100)
001	01100 (12_{10})	Load data stored at address 12_{10} (the # Z) into R1 (=00010000)
101	00000	Add R1 and R2, store in R2 (=00011100)
100	01011	Store data inside R2 at address 11_{10} (the # Y)
000	00000	stop

b. Given the contents of memory BEFORE the execution of the program, fill in the contents of memory AFTER the above machine instruction program has executed:

Address	Data (memory contents)
00001010 (10_{10})	00000011 X
00001011 (11_{10})	00001001 Y
00001100 (12_{10})	00010000 Z

BEFORE

Address	Data (memory contents)
00001010 (10_{10})	00000011 (not changed)
00001011 (11_{10})	00011100 (the sum)
00001100 (12_{10})	00010000 (not changed)

AFTER

 Name______________________________

Problem 7 — *The Rover (20 P)*

Student Ida Know has written the following lego program for the Mars Rover you built for this class. She did not comment it well at all. And some of the procedure names are boring and do not reflect the actual purpose of the code.

a. Comment her code in the areas that are outlined. Describe ***why*** a particular section of code exists rather than just describing what it does. (6p)

```
with Lego;
use Lego;

procedure Q4 is

   Left_Wheel  : constant Output_Port := Output_A;
   Right_Wheel : constant Output_Port := Output_C;
   Left_Rot    : constant Sensor_Port := Sensor_1;
   Right_Rot   : constant Sensor_Port := Sensor_3;
   Light       : constant Sensor_Port := Sensor_2;

   procedure P1 is
   begin
```

Initialize the power to the motors.

```
      --
      Output_Power(
         Output => Left_Wheel,
         Power  => Power_High);
      Output_Power(
         Output => Right_Wheel,
         Power  => Power_High);
```

Set the sensors to be treated as rotation sensors (which they are)

```
      --
      Config_Sensor(
         Sensor => Left_Rot,
         Config => Config_Rotation);
      Config_Sensor(
         Sensor => Right_Rot,
         Config => Config_Rotation);
```

The sensor on the Light port is configured as a light sensor.

```
      --
      Config_Sensor(
         Sensor => Light,
         Config => Config_Light);

      Output_Power(Left_Wheel,7);
      Output_Power(Right_Wheel,7);
      Clear_Sensor(Left_Rot);
      Clear_Sensor(Right_Rot);

   end P1;

   procedure Drive_Forward(Clicks:Integer) is
   begin

      --
      Clear_Sensor(Left_Rot);
      Clear_Sensor(Right_Rot);
      Output_On_Reverse(Left_Wheel);
      Output_On_Reverse(Right_Wheel);
```

```
      while (abs(Get_Sensor_Value(Left_Rot))<Clicks) loop
         Wait(10);
      end loop;

      Output_Off(Left_Wheel);
      Output_Off(Right_Wheel);

   end Drive_Forward;

   procedure P2 is
   begin
      --
      Output_On_Forward(Left_Wheel);
      Output_On_Reverse(Right_Wheel);

      Wait(1000);
      --
      Output_Off(Left_Wheel);
      Output_Off(Right_Wheel);


   end P2;

   Value : Integer;

begin
   --
   P1;
   Select_Display(Display_Sensor_2);
   loop
      Drive_Forward(1000);

      Wait(100);
      Value := Get_Sensor_Value(Light);

      P2;

      Wait(100);
      Value := Get_Sensor_Value(Light);
   end loop;
end;
```

Pivot turn left. (The direction can be derived from the drive forward function above).

Cut power to the wheels after 1000 time units.

Loop the following forever: drive forward, wait, read the light sensor, turn, wait read light sensor

In this comment, identify what the rover will do when this program executes

b. Rename the procedures (P1, P2) so they have more appropriate names that reflect what the procedure does for the lego rover (4 p)

P1___Initialize_Rover / Setup_Rover / similar…

P2___Turn_Rover(_Left) or similar…

 Name________________________________

c. I enjoyed the Lego portion of Unified C&P (please answer honestly). (2p)
   A. I agree
   B. I disagree
   C. Choose me and you get a zero
   D. I don't know/I don't understand.

Anything but B…

c. Short answer: Why should you call the lego procedure Clear_Sensor for the rotation sensor before doing a new rover maneuver? (2p)

If the sensor is not cleared the value it currently contains will be included in the maneuver! Another possible answer is the counter for the sensor will eventually overflow and reset to 0.

e. The Lego Rover ….(circle all that apply) (3p)

   1) runs Mission Critical Software.
   2) software is real time.
   3) is an embedded system.
   4) is an example of a Von Neuman Architecture.
   5) is used on the Pathfinder mission.
   6) is certified by NASA.
   7) has a separate floating point processor.

f. List 3 features of the standard Ada95 language that you cannot use with Ada Mindstorms and the Lego Rover (3p)

   1.____

   2.____

   3.____

There are numerous features that cannot be used, e.g., functions, more than 32+16 variables, multidimensional arrays, array initialization, floats, user defined types, most libraries, packages, etc.

 Name________________________________