Problem 1. Given the program below: **[15 Points]**

```
1. ------------------------------------------------------
2. -- Reading and modifying simple control statements
3. -- Programmer: Joe B
4. -- Date Created: October 12, 2004
5. ------------------------------------------------------
6. with Ada.Text_Io;
7.
8. procedure Demo_Simple_Control is
9.     My_Variable : Integer;
10.    My_Test     : Boolean;
11. begin
12.    My_Variable := 21;
13.
14.    if ((My_Variable rem 2)/=0) then
15.       Ada.Text_Io.Put_Line("Executing if successful");
16.       My_Test := True;
17.       My_Variable := My_Variable/2;
18.    else
19.       Ada.Text_Io.Put_Line("Executing if unsuccessful");
20.       My_Test := False;
21.    end if;
22.
23.    if My_Test then
24.       My_Variable := 1;
25.       if ((My_Variable/2) /=0) then
26.          Ada.Text_Io.Put_Line("Test and My_Variable satisifed");
27.       end if;
28.    else
29.       if ((My_Variable/2) = 0) then
30.          Ada.Text_Io.Put_Line("Nothing satisifed");
31.       end if;
32.    end if;
33. end Demo_Simple_Control;
34.
35. ------------------------------------------------------------
```

Part a. What is the output of the program? **[3 points]**

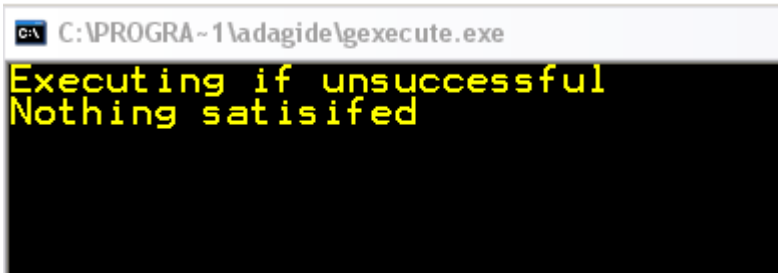
Executing if successful
Test and My_Variable satisfied

Part b. What are the values of My_Variable and My_Test when **[6 points]**

	My_Variable	My_Test
After Line 17 Executes	**10**	**T**

At the end of the program	**1**	**T**

Part c. For what value of My_Variable will you get the following output: **[6 points]**



When My_Variable has the value of: 0 (zero)

Problem 2 [10 points]

Write a recursive Ada95 function that gets a positive number as input and finds the sum of all positive odd integers from 1 to the largest odd number (<=input number).

The function is called from the main program using:

```
Sum := Recursive_Sum(Number);
```

For example

- `Sum:= Recursive_Sum(13)`, computes the sum of 1, 3, 5, 7, 9, 11, 13
- `Sum:= Recursive_Sum(12)`, computes the sum of 1, 3, 5, 7, 9, 11

The basic outline of the function is provided below.

```
function Recursive_Sum (Number : Integer) return Integer is
begin

      if Number <= 0 then
         return 0;
      else

         if Number mod 2 = 0 then
            return Recursive_Sum(Number-1);
         else
            return Number+Recursive_Sum(Number-2);
         end if;
      end if;

end Recursive_Sum;
```

Problem 3. [30 points]

Part a. Define an Ada95 record called `Per_Rec` with three fields: **[7 points]**

`Birth_Date`, of type integer

`Name`, of type string (assume a maximum of 10 characters)

`Grade`, of enumeration type with possible values (A, B, C)

Provide any additional type definitions you need for defining the fields.

```
type My_Enum is
       (A,
        B,
        C);

 type Per_Rec is
    record
       Birth_Date : Integer;
       Name       : String (1 .. 10);
       Grade      : My_Enum;
    end record;
```

Part b. Define an array to hold 5 records of type `Per_Rec` **[3 points]**

```
type Per_Rec_Array is array (1..5) of Per_Rec;

my_Per_Rec_Array : Per_Rec_Array;
```

Part c. **[20 points]**

Write an algorithm in the analysis format, to sort your array in descending order based on the `Birth_Date` field. If the `Birth_Date` fields are the same, then order the element in ascending order of the `Grade` field.
Assume that the array contains 5 valid records of type `Per_Rec`

Preconditions:
The input array contains 5 valid records of type Per_Rec
Temp is of type Per_Rec
Inputs:
Unsorted array

Outputs:
The sorted array has five records sorted in descending order based on the Birth_Date field. If the Birth_Date fields are the same, the records are sorted in ascending order based on the Grade field.

Postconditions:
The sorted array is returned to the calling program

Algorithm:

```
for I in 1 .. 4 loop
        for J in I+1 to 5 loop
                If my_Per_Rec_Array(I).Birth_Date < my_Per_Rec_Array(J).Birth_Date
                        Temp:= my_Per_Rec_Array(I)
                        my_Per_Rec_Array(I) := my_Per_Rec_Array(J)
                        my_Per_Rec_Array(J) := my_Per_Rec_Array(I)

                If my_Per_Rec_Array(I).Birth_Date = my_Per_Rec_Array(J).Birth_Date
                        If my_Per_Rec_Array(I).Grade > my_Per_Rec_Array(J).Grade
                                Temp:= my_Per_Rec_Array(I)
```

my_Per_Rec_Array(I) := my_Per_Rec_Array(J)

my_Per_Rec_Array(J) := my_Per_Rec_Array(I)

Problem 4. [15 Points]

Write a Pep/7 assembly language program to find the result of dividing an even number (from location num1) by two, using repeated subtraction. Store the result in location quotient.
Complete the template of the Pep/7 assembly code given below. Use additional memory locations or labels as needed.

```
; Program to demonstrate division by repeated subtraction
; Programmer:
; Date Created: October 15, 2004
; Date Last Modified:

BR Main;

; Data Segment
Num1:     .word d#40;
Quotient: .word d#0;

Temp:     .word d#0;



; Code Segment
Main:     LOADA num1, d; load num1 into accumulator

Loop:     BREQ Done;
          SUBA h#0002, i      ;subtract 2 from num1
          STOREA Temp, d      ;store the result of the
                              ;subtraction into temp
          LOADA Quotient, d   ;load the contents to
                              ;quotient into accumulator
          ADDA h#0001, i      ;increment the quotient by 1
          STOREA Quotient, d  ;store the value of the
```

```
                                ;quotient
          LOADA Temp, d         ;load the value of temp into
                                ;accumulator
          BR Loop               ;repeat the subtraction

Done:     DECO Quotient, d      ;display the contents of
                                ;quotient

          STOP
.END
```

Problem 5. **[20 Points]**

Part a. Convert the -28_{10} into 8 bit 2's complement notation. **[5 points]**

00011100 = 28
11100011 = 1's complement
+ 1 = addition of a 1
11100100 = -28_{10}

Part b. Convert $1AB_{16}$ into decimal **[5 points]**

$1AB_{16} = 1*16^2 + 10*16^1 + 11*16^0 = 427_{10}$

Part c. Convert 162.65625 into 32-bit floating point notation using: **[10 Points]**

- Scientific notation to represent the 23-bit mantissa
- Bias-127 notation to represent the 8-bit exponent

162 = 10100010
.65625 = .10101

162.65625 = 10100010.10101

Since scientific notation, the radix point has to be moved 7 steps to the left → 1.010001010101 (the numbers after the radix point goes into the mantissa field, the last 23 bits in the field below**)**

Bias-127 → 127

+ 7

134 = 10000110_2 (this is the number that goes into the exponent field**)**

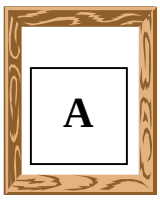
162.65625 is a positive number, so the sign bit below (bit furthest to the left) gets the value of: 0

0	1	0	0	0	0	1	1	0	0	1	0	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0

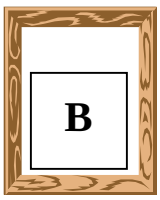
Problem 6 (10 points)

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.

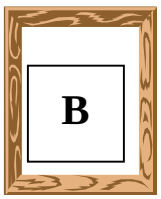
1. One of the following statements is true, which one?
   a. $ABBA_{16} = 1010\ 1011\ 1011\ 1010_2$
   b. $ABC_{16} = 1001\ 1010\ 1011_2$
   c. $101_{16} = 1000\ 0001_2$
   d. $101_{10} = 1000\ 0001_2$



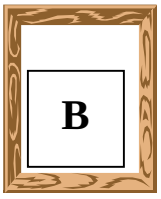
2. The ASCII code for characters ‘A’ and ‘a’ are the same.
   a. True
   b. False



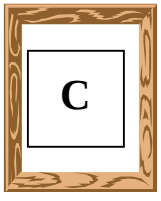
3. A string in Ada95 is
   a. A record of characters
   b. An array of characters
   c. An elementary data type



4. The Fetch-execute cycle
   a. Decode the instruction, get data if needed, fetch next instruction, execute the instruction
   b. Fetch the next instruction, decode the instruction, get data if needed, execute the instruction
   c. Fetch the next instruction, decode the instruction, execute the instruction, get data if needed



5. The von Neumann architecture is based around what principle?
   a. The Harvard Architecture



b. The instruction register contains the instruction that is being executed, and the program counter contains the address of the next instruction to be executed
c. Data and instructions are logically the same and can be stored in the same place