HW1
Due date: 10/17

1 Write a Fortran 95 program that reads in 10 integers in a line, sorts them by mergesort, and outputs the 10 sorted integers in a line. (30%)
Your program shall be organized as follows:

```
program sorting
! body of the program sorting

contains

recursive subroutine msort(a) ! a is an assumed-shape array
! body of the subroutine msort
end subroutine msort

subroutine merge(a)                ! a is an assumed-shape array
! This subroutine merges the already-sorted left half and right half of array a.
! The sizes of the two halves are determined by the subroutine msort.
end subroutine merge

end program sorting
```

Notes

1 Although the program tests on a 10-integer array, the **msort** subroutine shall be capable of sorting an array of arbitrary size.
2 The auxiliary array used in merging shall have the same size as the array being merged.
3 An array copy shall be done by a single assignment, i.e. without a loop.
4 The Boolean type is named **logical** in Fortran. It has the following constants and operators:
logical constants **.true. .false.**
logical operator **.and. .or. .not.**

Sample run

```
 Enter 10 integers
9 7 5 3 1 0 2 4 6 8
 The 10 integers in sorted order
0 1 2 3 4 5 6 7 8 9
```

2 Consider McCarthy's 91 function, where n is a signed integer.

$$f(n) = n - 10, \quad \text{if } n > 100$$
$$= f(f(n+11)), \quad \text{otherwise}$$

a) Write an imperative-style Javascript function to compute $f(n)$. (20%)
Hint: Use a variable, say *fs*, to record the number of times the function *f* remains to be called. Initialize *fs* with 1 and loop until *fs* becomes 0. Also, treat the parameter *n* as a variable and replace parameter passing by assignment.

b) Write a functional-style Javascript function to compute $f(n)$. (10%)

c) Compare a) and b) in terms of (1) programmer's productivity and (2) time and space complexities. (10%)

Notes

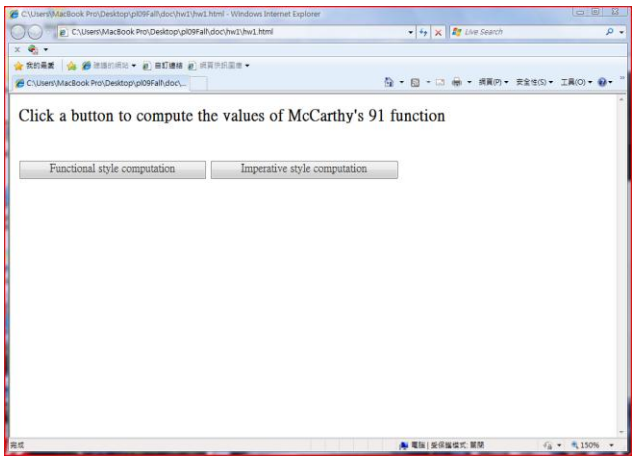
1 For simplicity, you may use the lecture example on html and Javascript as a framework. Your webpage shall have two buttons, one for invoking the imperative-style computation and the other the functional-style.

2 Your functional-style function shall count the number of times the function is called, and imperative-style function shall count the number of times the loop in it is executed.

3 As with the lecture example, your Javascript program shall respond to a non-number input with an "Illegal input" alert.
Note: If *s* is a string that cannot be converted to a number, the conversion of *s* to a number will yield an NaN (Not a Number). An NaN does not compare equal to any number. In particular, NaN ≠ NaN.

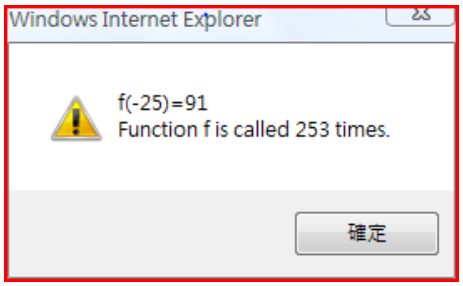
Sample run

Your webpage shall look like:

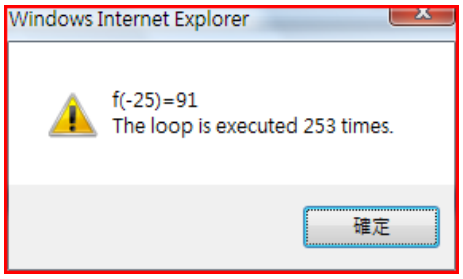


2 (Cont'd)

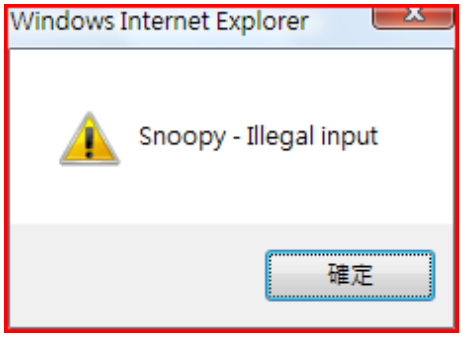
After clicking the button for function-style computation and entering -25, your program shall alert



After clicking the button for imperative-style computation and entering -25, your program shall alert



After clicking either button and entering Snoopy, your program shall alert



3 Write a Perl program that reads in a number of integers in a line, sorts the integers by bubble sort, and prints out the sorted integers in a line.
Note: Your program shall contain a subroutine named **`bsort`** that implements the bubble sort. (20%)

Sample run

```
Enter integers in a line: 7 3 9 1 5 8 4 2 6
After sorting: 1 2 3 4 5 6 7 8 9
```

4 Write a Perl program that reads in an integer $n \geq 0$ and computes the n^{th} Fibonacci number by recursion and memoization. That is, in the course of computing *fib*(*n*), you shall memoize already-computed Fibonacci numbers in a table to avoid recomputation.

For the purpose of this exercise, you are asked to create a hash table. The keys of the hash table are strings of the form "fib(0)", "fib(1)", "fib(2)", etc. The values corresponding to the preceding keys are 0, 1, and 1, respectively. (20%)

Sample run

```
Enter an integer>=0: 3
fib(3) = 2
Hash table created so far
fib(2) => 1
fib(3) => 2
fib(0) => 0
fib(1) => 1

Enter an integer>=0: 6
fib(6) = 8
Hash table created so far
fib(5) => 5
fib(2) => 1
fib(3) => 2
fib(0) => 0
fib(1) => 1
fib(4) => 3
fib(6) => 8

Enter an integer>=0: ^D
```