Department of Computer Science
University of Toronto
CSC108F – Midterm Test
Instructor: E. Fiume

Name: ____________________________________

Student Number: __________________________

TA (circle one): TK Anand A. Leganchuk S. Ju N. Shulman

- No aids allowed.
- Time allowed: 45 minutes.
- Read each question carefully and answer neatly in the space provided.
- Use the opposite sides of paper for rough work.

1	
2	
3	
4	
Total	

Next page: Question 1

1. [10 marks]
Simulate the following program and indicate what it would output. Then explain in words what the program computes. The expression **`floor(x)`** denotes the integer part of the real value **`x`**. So,

`floor(2.3)` is 2, and if **`y`** is 2, then **`floor(y/3)`** is 0

The expression **`x/m`** is a **`real`**-valued division, regardless of the numeric type of **`x`** or **`m`**.

```
var a,b,c: int
a := 0
b := 1
for i:1..10
    c := a+b
    put c, (c/10 - floor(c/10))*10   % don't let this scare you!
    a := b
    b := c
end for
```

The output is:

11
22
33
55
88
133
211
344
555
899

Now explain what the program does.

The program internally creates a sequence of 10 numbers. Each element in that sequence is created by summing the two previous elements in the sequence, starting with two initial values of 0 and 1. The output is each of the ten elements of that sequence with the rightmost digit duplicated.

Next page: Question 2

2. [10 marks]
The program below is supposed to input a user’s full name on one line and then input five integer grades. It should compute the average of these five grades. The output should be of the form

```
<person’s name>: <average>.
```

So, on input

```
Eugene Fiume
75
80
85
60
72
```

the output should be

```
Eugene Fiume: 74.4.
```

Unfortunately, the person writing the program below has not taken CSC108F, and so there are many syntactic, semantic and stylistic errors. Identify at least ten errors by (neatly) circling them (including at least one example of each kind of error). Then correct each error or add code as needed to get the program to work correctly. Add explanations as you wish to the right of the code.

`var name: int`	var name: string	(1)
`var average: int`	var average: real	(2)
`var count: int`	const count: int := 5	(3,4)
`var mark, sum: int`	var mark, sum: int	
`var i: int`	% index variables aren’t declared	(5)
`% get name`	get name: * % remove comment & fix length	(6,7)
`sum = 1`	sum := 0 % bad init and assignment	(8,9)
`for i:1..count`	for i:1..count	
`    get mark`	get mark	
`    sum := sum + mark`	sum := sum + mark	
	end for % missing end for	(10)
`put "name"`	put name, ": " ..	(11)
`average := sum/5`	average := sum/count	(12)
`put average`	put average, "."	(13)

Next page: Question 3

3. [10 marks]
Given two strings **s1** and **s2**, and an integer **i**, construct and output a new string **s3** that contains the result of inserting **s2** into **s1** beginning at character position **i** of **s1**. Once all of **s2** has been inserted, the remainder of **s1** should be appended. For example, if **s1** is "Heo", **s2** is "ll", and **i** is 3, then after the operation, **s3** should be "Hello". Your code should cope with the possibility that **i** is an inappropriate value. When **i** is inappropriate, your code should create the most reasonable string as a result (you decide what that means and comment on it). Add your solution to this program fragment:

```
var s1,s2,s3: string
var i: int
put "Enter two strings:"
get s1, s2
put "Enter a positive integer:"
get i
%
% It's all yours now.
%
```

```
%
% So, the two erroneous cases to check are that i <= 0 or that i > length(s1).
% In the first case, we shall assume the user wants s2 appended before s1.
% In the second case, we append s2 past s1, so change i to do this.
%
if i <= 0 then
        i := 1
end if

if i > length(s1) then
        i := length(s1)+1
end if

s3 := s1(1..i-1) + s2 + s1(i..*)
put s3
```

Next page: Question 4

4. [10 marks]
Suppose you have an array **`A`** containing **`n`** integers. The user inputs a value **`k`** that is between 1 and **`n`**, and you are to output the **`k`**th largest number in the list. If, for example, **`A`** contains

`7, 2, 4, 5, 1, 5`

and **`k`** is 3, then our output should be the third-largest number in the list, namely 4. Now, suppose you have at your disposal two fragments of Turing code: one will sort **`A`** for you in descending order, putting the result back in **`A`**, and the other will find the maximum of **`A`** giving you the index **`i`** for which **`A(i)`** is a maximum. We will call the first fragment **`SORT`** and the second **`MAX`**. Given the code below to start you off, use only one of these two fragments to find and output the **`k`**th largest number in **`A`**. The question is asking you to *use* one of **`SORT`** or **`MAX`**, not write them. Add your solution to the following code fragment.

```
const N := 1000
var numbers: array 1..N of int
var int: i,j,k,n     % you may not need j
%
% Assume the user has now input A(1) through to A(n), the integer k
% and that both k and n are valid.  Add your code using MAX or SORT here.
SORT
var prev: int := A(1)
j := 1
i := 2
loop
        exit when i>n or j=k
        if A(i) not= prev then
                prev := A(i)          % unique maximum
                j := j+1
        end if
        i := i+1
end loop

if j=k then
        put k, "th largest element is ", A(i-1), "."
else
        put "there is no ", k, "th largest element."
end if
```

For the fragment **`MAX`** or **`SORT`** that you *did not* use, briefly outline in *English* what you would do if you had to use it instead.

To use MAX, I would repeatedly invoke MAX find the maximum, and determine the k-th unique maximum by keeping track of my previous maximum in A. I would stop upon finding the k-th unique maximum and output the result. To ensure that I don't keep finding the same maximum repeatedly, I have to remove the maximal element from A by copying all of A except for that element to another array, and then copying it back to A, and decreasing my record of the number of elements in A.

End of test and solutions.