Name_____ Student ID_____ Department/Year_____

Final Examination

Introduction to Computer Science
Class#: EE1003, Session#: 03
Spring 2017

15:30-17:10 Wednesday
June 21, 2017

Prohibited

1. You are not allowed to write down the answers using pencils. Use only black- or blue-inked pens.
2. You are not allowed to read books or any references not on the question sheets.
3. You are not allowed to use calculators or electronic devices in any form.
4. You are not allowed to use extra sheets of papers.
5. You are not allowed to have any oral, visual, gesture exchange about the exam questions or answers during the exam.

Cautions

1. Check if you get **18** pages (including this title page), **13** questions.
2. Write your name (in Chinese), student ID, and department/year down on top of the cover page.
3. There are in total **101** points to earn. You have **100 minutes** to answer the questions. Skim through all questions and start from the questions you feel more confident with.
4. You are allowed to use **English only** to answer the questions. Misspelling and grammar errors will be tolerated, but you want to make sure with those errors your answers will still make sense.
5. If you have any extra-exam emergency or problem regarding the exam questions, raise your hand quietly. The exam administrator will approach you and deal with the problem.

1. The Internet protocols are classified in layers. Identify the layer these 5 Internet protocols, TCP, HTTP, FTP, Ethernet, RIP, belong to. (5%)
   (a) Application Layer
   (b) Transport Layer
   (c) Network Layer
   (d) Link Layer

Sample Solution:

TCP (b), HTTP (a), FTP (a), Ethernet (d), RIP (c)

2. Which of the following are functions of a Web browser? (5%)
   (a) HTTP client
   (b) HTTP server
   (c) HTML interpreter
   (d) HTML compiler

Sample Solution:

(a), (c)

3. Given the following statements about the public-key encryption system, identify the correct ones. (5%)
   (a) a message encrypted by the public key can be decrypted by the public key
   (b) a message encrypted by the private key can be decrypted by the public key
   (c) a message that can be decrypted by the public key suggests authentication
   (d) a message that can be decrypted by the private key suggests authentication

Sample Solution:

(b), (c)

4. Consider the following algorithm.

```
F[0] <- 0;
F[1] <- 1;
for (i=2; i<=9; i++) do
    (F[i] <- F[i-1] + F[i-2];)
print F[9];
```

(a) Write the print-out of the program. (5%)
(b) Rewrite the algorithm using a while-do loop. (5%)
(c) Rewrite the algorithm using a repeat-until loop. (5%)
(d) Rewrite the algorithm using a recursive structure. (5%)

Sample Solution:

(a) 34

```
F[0]=0, F[1]=1, F[2]=1,  F[3]=2,  F[4]=3,
F[5]=5, F[6]=8, F[7]=13, F[8]=21, F[9]=34
```

(b)

```
F[0] <- 0;
F[1] <- 1;
i=2;
while (i <= 9) do
      (F[i] <- F[i-1] + F[i-2];
       i++;)
print F[9];
```

(c)

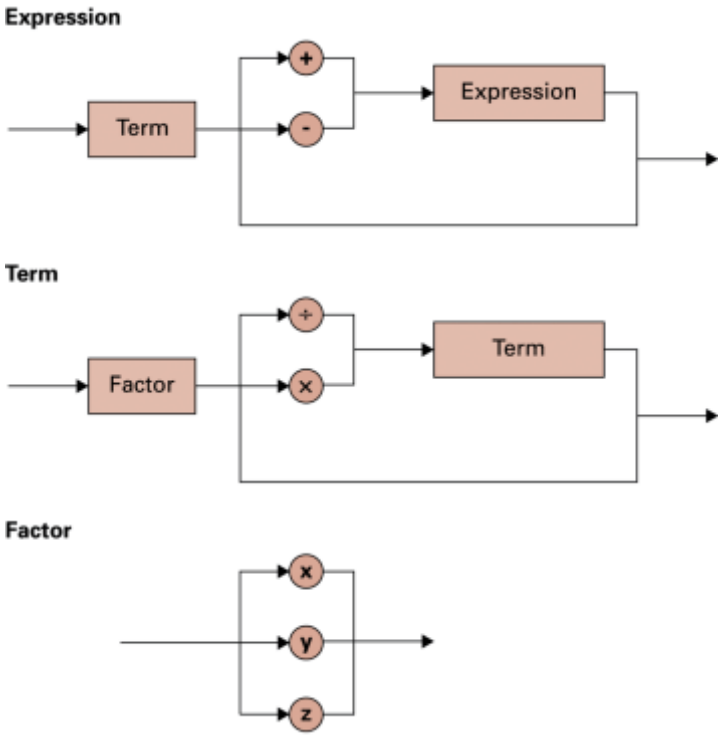
```
F[0] <- 0;
F[1] <- 1;
i=2;
repeat
      (F[i] <- F[i-1] + F[i-2];
       i++;)
until (i > 9)
print F[9];
```

(d)

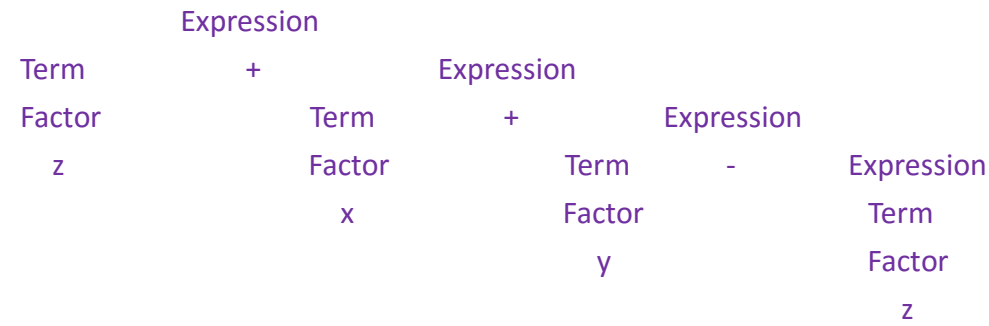
```
int function Fibonacci (i)
      (if (i==0) then
         return 0;
      if (i==1) then
         return 1;
      if (i>=2) then
         return Fibonacci(i-1)+Fibonacci(i-2);)

print Fibonacci (9);
```

5. Given the syntax rules below. Draw the parse tree for z+x+y-z (5%)

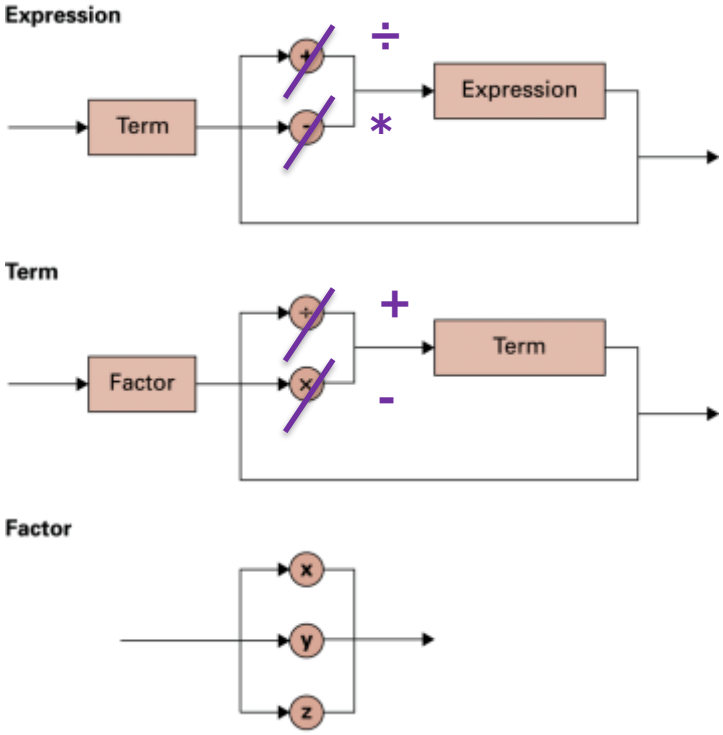


Sample Solution:



6. Continue from 5. The syntax rules give precedence to multiplication and division. Redefine the syntax rules such that the precedence is given to addition and subtraction instead. (5%)

Sample Solution:



7. Find the resolution of multiple statements:
   (a) $(P \vee Q)$ and $(\neg Q \vee R)$ and $\neg R$ (5%)
   (b) $(P \vee Q)$ and $(\neg Q \vee R)$ and $\neg R$ and $(\neg P \vee \neg S)$ and $(S \vee \neg T)$ (5%)

Sample Solution:

(a) Many possible resolvents. The key resolvents are: P, ¬Q

(b) Many possible resolvents. The key resolvents are: ¬T, ¬S, P

8. There is a sea of problems that computer scientists are curious about. The *Church-Turing Thesis* states boldly that:

   "Computable and *Turing Computable* problems are one and the same."

   To prove the *Church-Turing Thesis*, one will need to prove that (1) Any *Computable* problem is *Turing Computable*, and (2) Any *Turing Computable* problem is *Computable*. Proving (1) is not trivial and that's when the Bare Bone language is introduced. Before we attempt to prove the Church-Turing thesis through Problem 9-11, define these basic terms first.

   (a) What are '*Computable*' problems? (2%)
   (b) What are '*Turing Computable*' problems? (2%)
   (c) What are '*Bare Bone Computable*' problems? (2%)

Sample Solution:

(a) Computable problem – a problem for which there exists an algorithmic solution. Therefore, there exists a solution consisting of a set of ordered, unambiguous steps that any programmer with the help of a general programming language can implemented into a functional program that solves the problem.
(b) Turing Computable problem – a problem for which there exists a Turing machine solution
(c) Bare Bone Computable problem – a problem for which there exists a solution implemented in the Bare Bone language

9. Continue from 8. Proving (1) is equivalent of proving these two statements together: (3) Any *Computable* problem is *Bare Bone Computable* and (4) Any *Bare Bone Computable* problem is *Turing Computable*. Proving (3) is tedious. One will need to show that for each statement in a general programming language, there exists an equivalent Bare Bone program.

   We've shown in the class and the old exams that the 'assignment' statement, some of the 'arithmetic' operations, and the 'if-then-else' statement commonly seen in a general programming language can be represented using the Bare Bone language. Let's try to enrich the set of statements in general languages that can be written in Bare Bone to the 'for-loop' statement.

   Assume X is a positive integer. Show how the following 'for-loop' statement could be simulated in the Bare Bone language. (10%)

```
for (name1=1; name1<=X; name1++)
    name2+=name1;
```

Sample Solution:

There are many possibilities. Here's just one example:

```
clear name1
clear name2
clear auxX
clear auxName1

while (X not 0)
    incr name1              ---> start from name1 = 1
    while (name1 not 0)     ---> name2+=name1
        incr name2
        decr name1
        incr auxName1
    while (auxName1 not 0) ---> restore name1
        incr name1
        decr auxName1
    decr X
```

```
    incr auxX

while (auxX not 0)          ---> restore X
    incr X
    decr auxX
```

10. Continue from 9. Proving (4) is less tedious. One just need to show for each of the 4 statements in Bare Bone, there exists a Turing machine equivalent.

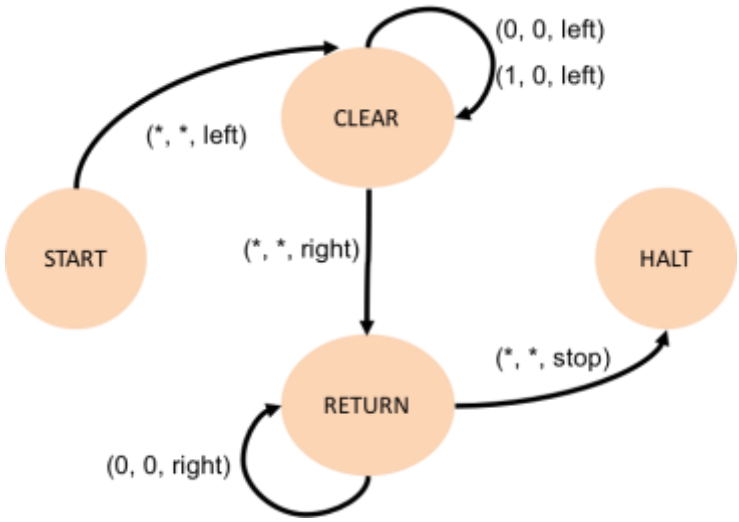
    We've discovered in the class that the Turing machine equivalent of the 'incr' and 'decr' statements. Let's try to enrich the set of Turing machines to the 'clear' statement.

    Use the same notation as the 'incr' and 'decr' Turing machine we've used in the class. That is, on the tape, a number is represented by a sequence of 0 or 1 and bounded by a pair of '*'s. The read/write head starts off reading the '*' on the right. Generate the state machine for the 'clear' statement, where all bits of a sequence are reset to 0, as the state machine comes to halt. (10%)

Sample Solution:

The solution can be either in the table for the graphic form.

Current state	Current value	Value to write	Direction to move	Next state
START	*	*	Left	CLEAR
CLEAR	0	0	Left	CLEAR
CLEAR	1	0	Left	CLEAR
CLEAR	*	*	Right	RETURN
RETURN	0	0	Right	RETURN
RETURN	*	*	No move	HALT



11. Continue from 8. Proving (2) is considered trivial. How would you argue a *Turing Computable* problem is *Computable*? (5%)

Sample Solution:

Given a Turing machine, one can represent (1) the states as ‘procedures’, (2) inside the procedure, the transitions as a number ‘if-then-else’ statements, and (3) the content on the tape slots as variables.

12. Draw the search tree to solve the eight-puzzle from the following start state.

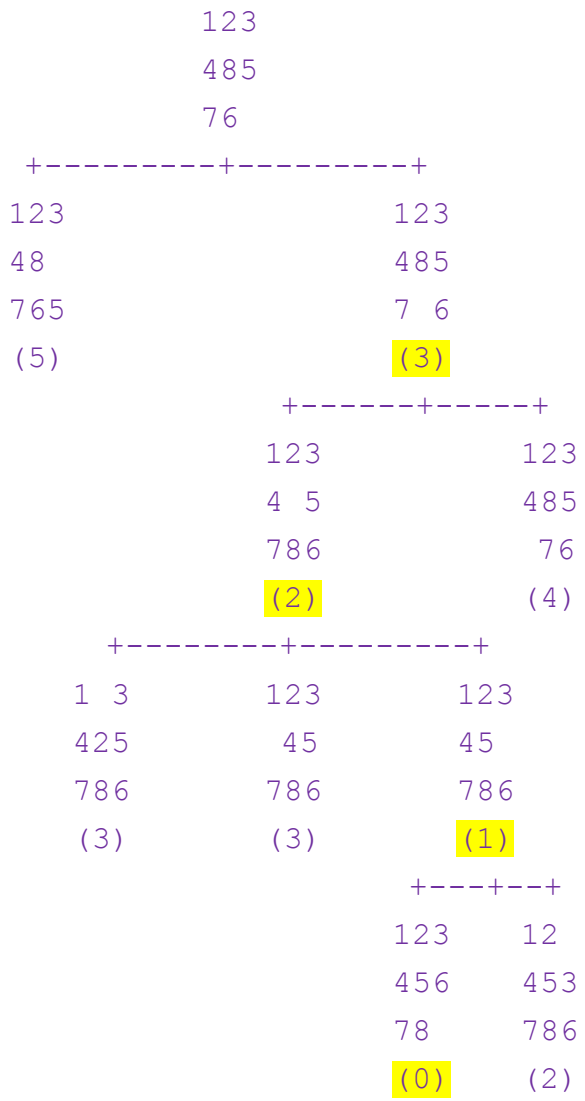
```
123
485
76
```

(a) Use the heuristic search as defined in the class. (5%)

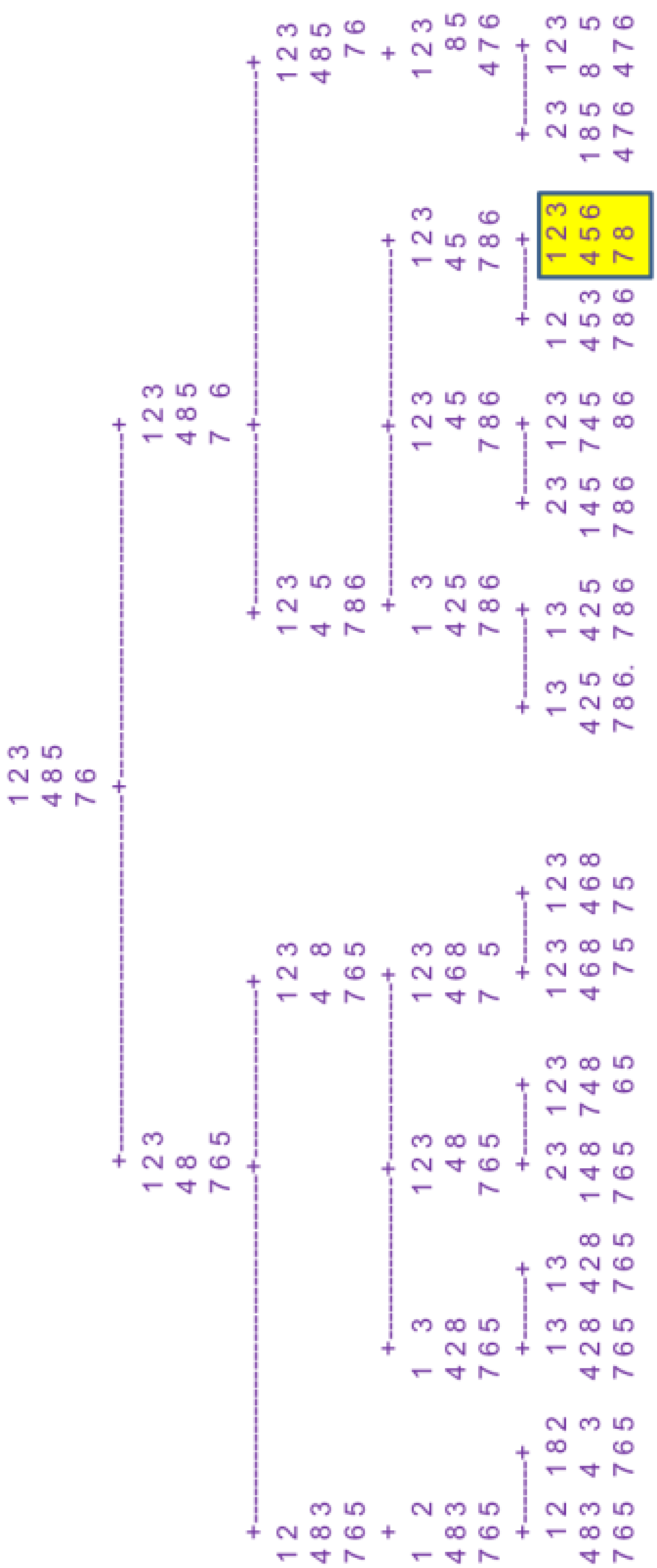
(b) User the breadth-first search without any heuristics. (5%)

Sample Solution:

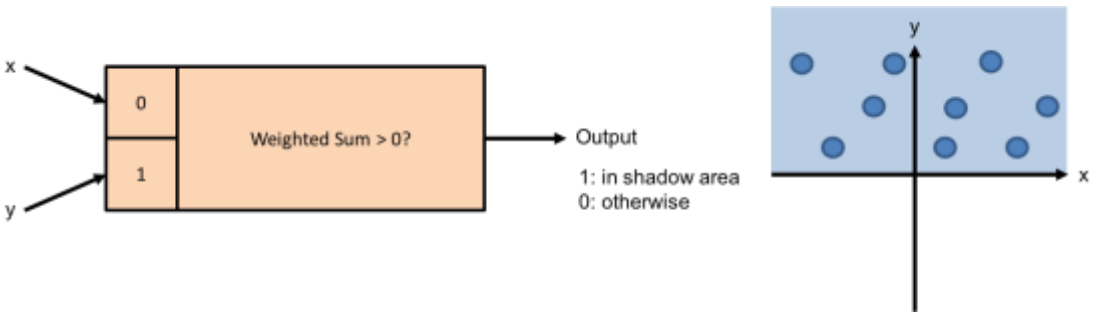
(a)



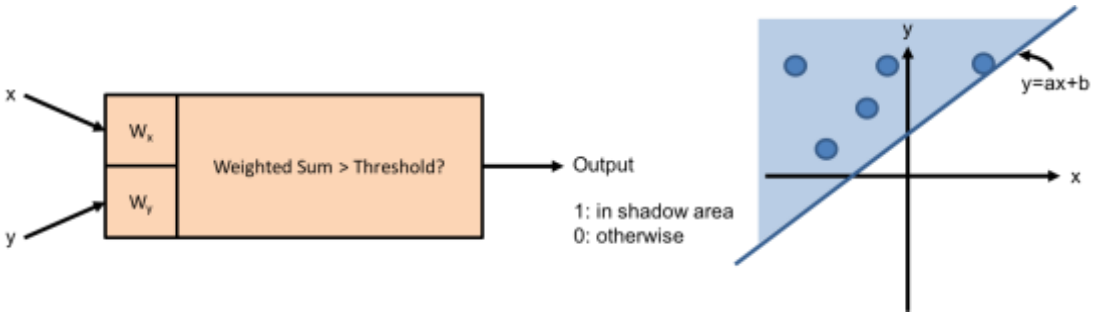
(b)



13. The simple artificial neural network below takes the coordinate (x, y) of an input, and outputs 1 if the input falls in the 'y>0' area, the space in light shadow.



Now, try to reconfigure the artificial neural network such that it detects inputs falling in the shadowed 'y>ax+b' area instead.



How should you configure the W_x, W_y and Threshold? (5%)

Sample Solution:

To ensure y-ax>b (equivalent of y>ax+b)

W_x= -a

W_y= 1

Threshold= b