UNIVERSITÀ DEGLI STUDI "ROMA TRE"
CORSO DI STUDI IN SCIENZE COMPUTAZIONALI
IN490 - LINGUAGGI DI PROGRAMMAZIONE – A.A. 2017-2018

M. PEDICINI

ESAME DEL 20/06/2018 – TEMPO 3H00

COGNOME _____ NOME _____ MATRICOLA _____

Esercizio 1. *One Instruction Set Computer (OISC), significa una macchina M con un linguaggio di una sola istruzione: un esempio di OISC è la macchina $M_{\texttt{subleq}}$. La macchina ha una memoria che contiene interi (relativi) e una sola istruzione* `subleq`*, l'istruzione ha tre argomenti interi, poichè non ci sono altre istruzioni è possibile omettere l'*op-code *nel compilato.*

L'esecuzione dell'istruzione `subleq a b c` *produce la tripla* `a b-a c` *e l'esecuzione salta all'indirizzo* `c` *se* `b-a` *è un valore minore o uguale a zero, di fatto l'istruzione* `subleq` *viene applicata tramite indirizzamento indiretto all'indirizzo di memoria dell'istruzione corrente e alle due locazioni successive; dunque, eventualmente, viene modifica la cella di memoria il cui indirizzo è contenuto nella cella successiva a quella dell'istruzione corrente.*

Per maggiore chiarezza diamo qui lo pseudo-codice di un interprete per un programma contenuto nell'array di memoria `memory[]`*:*

```
int memory[], program_counter, a, b, c
 program_counter = 0
 while (program_counter >= 0):
     a = memory[program_counter]
     b = memory[program_counter+1]
     c = memory[program_counter+2]

     memory[b] = memory[b] - memory[a]
     if (memory[b] > 0):
         program_counter += 3
     else:
         program_counter = c
```

Un programma in linguaggio macchina è dunque dato (come al solito) da una lista di interi da caricare in memoria che però non contiene opcodes*.*

Ad esempio il programma che ha in memoria i valori `2 1 0` *rappresenta un loop infinito che decrementa con passo* `2` *la seconda posizione in memoria.*

(1) *Scrivere un programma che esegue un loop infinito che incrementa con passo* `1` *la seconda posizione della memoria.*
(2) *tradurre l'istruzione* `JMP c` *(salto incondizionato) in subleq;*
(3) *tradurre l'istruzione* `RST c` *che pone a zero la cella di memoria* `c` *in subleq;*
(4) *sia* `Z` *l'indirizzo di una cella di memoria ausiliaria nella quale appoggiare valori temporaneamente e nella quale faremo in modo di mantenere il valore zero: tradurre l'istruzione* `ADD a b` *(somma dei valori contenuti) in subleq;*
    [hint: in 3 istruzioni – utilizzare la cella `Z` *per salvare l'opposto di* `a`*]*
(5) *tradurre l'istruzione* `MOV a b` *che "copia" il valore contenuto all'indirizzo* `a` *nella cella di indirizzo* `b`*;*
    [hint: azzerare `b` *e poi utilizzare la* `ADD` *]*

(6) *tradurre l'istruzione* `BEQ b c` *che salta alla cella* `c` *se il valore contenuto all'indirizzo* `b` *è zero;*

(7) *Tradurre le istruzioni per la macchina RAM* `LOAD`, `STORE`, `JUMP`, `JTZ`, `ADD`, `SUB`, `MULT` *nel linguaggio subleq.*

[hint: fissare una locazione per l'accumulatore e utilizzare le istruzioni tradotte ai punti precedenti come linguaggio intermedio]

Esercizio 2. (1) *Quando una grammatica si dice ambigua ? Fornire la definizione.*

(2) *Utilizzando la nozione di ambiguità appena fornita, mostrare che la grammatica*

$$G = (\{a, b\}, \{S, T\}, S, R)$$

con

$$R = \begin{cases} S \to SaS|TbT|T \\ T \to TaT|S|a \end{cases}$$

è ambigua;

(3) *fornire una grammatica G' non ambigua che genera lo stesso linguaggio, ovvero per cui $L(G) = L(G')$.*

Esercizio 3. (1) *Si descriva il costrutto delle* eccezioni *in un linguaggio di programmazione.*

(2) *Si descriva l'output stampato dal seguente frammento in cui il passaggio dei parametri avviene per* riferimento*:*

```
 1 int x = 2;
 2 int w = 6;
 3 int z = 11;
 4
 5 int f (int y){
 6 y = 21;
 7 throw E;
 8 return (x++)+y;
 9 }
10
11 int g (int y){
12  y = 100;
13  try { throw E; } catch E {}
14  return (x++) + y;
15 }
16
17 try { f(w); } catch E {} print(x,w,z);
18 z = g(w);
19 print(x,w,z);
```