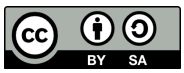
Esercizi di Fondamenti di Informatica A

Corso di laurea in Ingegneria Biomedica e
Ingegneria Elettronica per l'energia e l'informazione
Università di Bologna

Anno Accademico 2017/2018

Ultimo aggiornamento: 23 giugno 2018



Commenti
Suggerimenti e correzioni sono benvenuti: scrivetemi all'indirizzo moreno.marzolla@unibo.it.

Ultima versione
La versione più recente di questo documento (in formato PDF e LibreOffice) si trova all'indirizzo http://www.moreno.marzolla.name/

Revisioni

- Aggiornamento del 12/5/2018: Aggiunti esercizi su asserzioni e invarianti
- Aggiornamento del 17/5/2018: Aggiunti ulteriori esercizi su asserzioni e invarianti

Domande

Esercizio 1 Macchine di Turing

Consideriamo una Macchina di Turing (MdT) il cui alfabeto sia composto dai simboli {*blank*, 1}, e i cui stati siano {q0, *halt*} con q0 stato iniziale. La tabella delle istruzioni della macchina è la seguente:

Stato corrente	Simbolo corrente	Nuovo simbolo	Nuovo stato	Spostamento
q0	1	1	q0	right
q0	*blank*	1	halt	---

Ricordiamo che in una MdT si assume che il nastro sia inizialmente vuoto, tranne al più un sottoinsieme *finito* di celle che possono contenere simboli dell'alfabeto diversi da *blank*. Per ciascuna delle seguenti affermazioni relativa alla macchina di cui sopra, dire se è vera (V) o falsa (F)

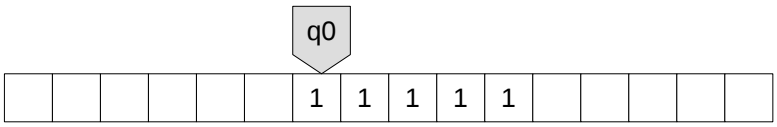
V *F* Se la macchina viene fatta operare su un nastro in cui tutte le caselle sono *blank*, allora scrive un unico simbolo 1 e poi termina;

V *F* La macchina termina sempre, qualunque sia il contenuto iniziale del nastro e la posizione iniziale della testina di lettura/scrittura;

V *F* La macchina modifica sempre almeno una cella del nastro (cioè modifica il contenuto di almeno una cella del nastro, scrivendo un simbolo diverso da quello precedente), qualunque sia il contenuto iniziale del nostro e la posizione iniziale della testina di lettura/scrittura.

Esercizio 2 Macchine di Turing

Progettare una Macchina di Turing (MdT) il cui alfabeto sia composto dai simboli {1, *blank*}, e i cui stati siano {q0, *halt*} con q0 stato iniziale. Il nastro della MdT contiene inizialmente una sequenza di celle adiacenti contenenti 1, e la testina di lettura/scrittura è posizionata sulla prima cella a sinistra contenente un 1, come nella figura seguente:



La MdT deve cancellare tutti gli 1 presenti sul nastro, sostituendoli con il carattere *blank*, e poi fermarsi. La MdT deve funzionare anche quando il nastro è inizialmente vuoto (cioè nel caso in cui non siano presenti celle inizializzate a 1). Completare la tabella delle istruzioni seguente in modo da realizzare quanto richiesto

Risposta (compilare la tabella):

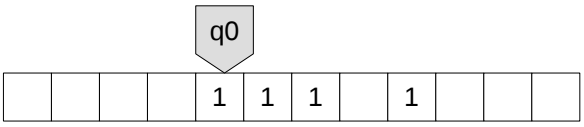
Stato corrente	Simbolo corrente	Nuovo simbolo	Nuovo stato	Spostamento
q0	1			
q0	*blank*			

Esercizio 3 Macchine di Turing

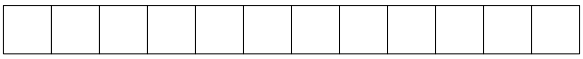
Consideriamo una Macchina di Turing (MdT) il cui alfabeto sia composto dai simboli {*blank*, 1}, e i cui stati siano {q0, *halt*} con q0 stato iniziale. La tabella delle istruzioni della macchina è la seguente:

Stato corrente	Simbolo corrente	Nuovo simbolo	Nuovo stato	Spostamento
q0	1	*blank*	q0	right
q0	*blank*	1	halt	---

Supponendo che la MdT venga messa in esecuzione su un nastro avente la configurazione seguente, con la testina di lettura/scrittura inizialmente posizionata come indicato



disegnare qui sotto il contenuto del nastro al termine dell'esecuzione del programma



Esercizio 4 Circuito

Disegnare un circuito logico, utilizzando esclusivamente porte di tipo AND, OR e NOT (è possibile usare un numero qualsiasi di porte di ciascun tipo, anche nessuna) che abbia un singolo input *A* e produca un singolo output *R* che abbia sempre valore 0. In altri termini, il circuito deve realizzare la seguente tabella di verità

A	*R*
0	0
1	0

Attenzione: non è ammesso l'uso di ulteriori input costanti; in altre parole, l'unico segnale che il circuito può manipolare è *A* (e ogni ulteriore segnale derivabile da *A*)

Risposta:

Esercizio 5 Tabella di verità

Compilare la tabella di verità dela seguente espressione booleana:

$$R = (A \textbf{ AND } B) \textbf{ OR } (\textbf{NOT } A)$$

Risposta (compilare la tabella):

A	*B*	*R*
0	0	
0	1	
1	0	
1	1	

Esercizio 6 Codifica binaria

Si considerino due numeri interi A e B rappresentati in complemento a due con 8 bit, con $A = 00110110_{2C}$ $B = 10001100_{2C}$. Per ciascuna delle seguenti affermazioni, dire se è vera (V) o falsa (F)

V *F* A è un numero positivo

V *F* B è un numero positivo

V *F* Il valore esatto della somma $(A + B)$ è rappresentabile in complemento a due con 8 bit

V *F* A vale 87

V *F* A è minore di B

Esercizio 7 Codifica binaria

Si consideri la codifica di numeri interi in complemento a 2 con soli 5 bit, allora:

V *F* Il più grande intero codificabile è 32

V *F* Il più piccolo intero codificabile è -16

V *F* La sequenza di bit 01101_{2C} codifica un numero positivo

V *F* La sequenza di bit 10011_{2C} codifica il valore -13

V *F* La somma dei numeri (13 + 7) è rappresentabile in complemento a 2 usando solo 5 bit.

Esercizio 8 Codifica dell'informazione

Supponiamo di codificare una immagine bitmap a colori con le seguenti caratteristiche:

- risoluzione 320×200 pixel
- il colore di ciascun pixel viene rappresentato usando 5 bit per la componente rossa, 5 bit per la componente verde e 5 bit per la componente blu

Quanti bit sono necessari per codificare l'immagine?

Risposta (non è necessario svolgere i calcoli, basta indicare l'espressione):

Esercizio 9 Codifica dell'informazione

Supponiamo che un certo alfabeto sia composto da un totale di 47 caratteri (inclusi numeri e segni di punteggiatura). Quale è il numero minimo di bit con cui è possibile assegnare a ciascuno dei 47 caratteri una codifica univoca?

Selezionare una singola risposta tra le seguenti:

- 4 bit
- 6 bit
- 8 bit
- Nessuna delle precedenti

Esercizio 10 Codifica dell'informazione

Nella codifica ASCII a 8 bit dei caratteri, le lettere A—Z (maiuscole) sono codificate con gli interi 65—90 ('A' = 65, 'B' = 66, 'C' = 67, … 'Z' = 90). Quale parola è rappresentata dai caratteri seguenti, i cui codici ASCII sono indicati in notazione binaria? Completare la tabella seguente, indicando il valore decimale che corrisponde a ciascun valore bianario, e quindi il carattere.

ASCII binario	*ASCII decimale*	*Carattere*
0100 0011		
0100 1001		
0100 0001		
0100 1111		

Esercizio 11 Banda e latenza

Un'auto percorre una strada lunga 32 Km (32000 m) alla velocità costante di 16 m/s (corrispondenti a circa 57.6 km/h) trasportando una memory card della capacità di 32 GB; per semplicità si assuma che 1GB corrisponda a 1000000000 B. Determinare la banda (in Byte al secondo) e la latenza (in secondi) del sistema.

Risposta (banda):

Risposta (latenza):

Esercizio 12 Reti

In cosa differiscono le reti a commutazione di circuito e a commutazione di pacchetto?

Risposta (massimo 5 righe):

Esercizio 13 Crittografia

In cosa differiscono la crittografia simmetrica e la crittografia asimmetrica?

Risposta (massimo 5 righe):

Esercizio 14 Crittografia

Supponiamo che *K+* e *K-* siano una coppia (chiave pubblica, chiave privata) generata da un sistema di crittografia a chiave pubblica. Allora

V *F* Un messaggio *M* codificato con *K+* deve essere decodificato con la stessa chiave *K+*

V *F* E' estremamente “difficile” derivare una delle due chiavi, anche se si conosce l'altra

V *F* E' possibile codificare un messaggio *M* utilizzando *K*-, e in tal caso il messaggio cifrato può essere decodificato solo usando *K*+

V *F* *K*+ e *K*- devono coincidere, ossia essere la stessa chiave

Esercizio 15 Crittografia

Una certa parola è codificata con il “cifrario di Cesare”, in cui ogni lettera dell'alfabeto viene cifrata con la lettera immediatamente successiva. Quindi la parola `CIAO` viene cifrata come `DJBP`. A quale parola corrisponde la stringa cifrata `JOGPSNBUJDB` ?

Risposta:

Esercizio 16 Grammatiche

Scrivere una grammatica BNF in grado di generare tutte e sole le stringhe che rappresentano gli otto punti cardinali di una rosa dei venti come la seguente:

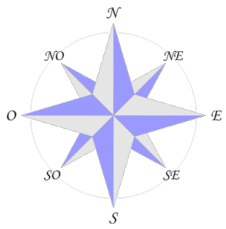


immagine di ElfQrin - Opera propria, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=15810334

In altre parole, la grammatica deve generare tutte e sole le stringhe {“N”, “S”, “E”, “O”, “NE”, “SE”, “NO”, “SO”}. La grammatica DEVE fare uso di almeno due simboli non terminali, ivi compreso il simbolo iniziale. In altri termini, la grammatica seguente, sebbene tecnicamente corretta, NON sarà accettata come risposta:

<Rosa> ::= “N” | “S” | “E” | “O” | “NE” | “SE” | “NO” | “SO”

Risposta:

Esercizio 17 Grammatiche

Si consideri la grammatica BNF seguente

<A> ::= ε | 00<A>

dove <A> rappresenta l'unico simbolo non terminale (e anche il simbolo iniziale della grammatica), e 0 rappresenta l'unico simbolo terminale (omettiamo le virgolette per semplicità). Allora:

V *F* La grammatica genera lo stesso insieme di parole della grammatica <A> ::= ε | 0<A>0

V *F* La grammatica genera un insieme finito di parole

V *F* La grammatica può generare parole con un numero dispari di caratteri “0”

Esercizio 18 Grammatiche

Scrivere una grammatica BNF in grado di generare tutte e sole le sequenze binarie, di qualsiasi lunghezza purché non vuote, che rappresentano valori dispari se interpretate come numeri decimali non in complemento a due. Ad esempio, la grammatica deve poter generare sequenze binarie come 1, 001001, 1111111001, 00000000011, e così via.

Risposta:

Esercizio 19 Aritmetica dei puntatori in C

Consideriamo il seguente frammento di codice in linguaggio C:

```
int arr[] = {183, 22, -1, 14};
int* p = &arr[0];
int* q = p+1;
int r;
*q = 97;
r = arr[1];
```

Quale è il valore della variabile `r` al termine del frammento di codice? Selezionare una singola risposta tra le seguenti

- `22`
- `97`
- `183`
- Indefinito
- Nessuno dei precedenti

Esercizio 20 Aritmetica dei puntatori in C

Consideriamo il seguente frammento di codice in linguaggio C:

```
int v = 12;
int w;
int* p = &v;
int* q = &w;
p = q;
*q = 47;
```

Allora, al termine dell'esecuzione del frammento di codice sopra:

V *F* la variabile `v` ha valore indefinito

V *F* I puntatori `p` e `q` puntano alla stessa locazione di memoria

V *F* la variabile `w` ha valore indefinito

V *F* Le variabili `v` e `w` hanno lo stesso valore

Esercizio 21 Algoritmi

L'algoritmo di ricerca sequenziale:

V *F* Funziona correttamente solo se la lista dei valori in cui effettuare la ricerca è ordinata in senso crescente

V *F* Nel caso *migliore* termina dopo aver esaminato almeno metà degli elementi della lista

V *F* Nel caso *peggiore* termina dopo aver esaminato tutti gli elementi della lista

V *F* Deve esaminare tutti gli elementi della lista prima di poter concludere che il valore cercato non è presente

Esercizio 22 Macchine di Turing

Consideriamo una Macchina di Turing (MdT) il cui alfabeto sia composto dai simboli {*blank*, 0, 1}, e i cui stati siano {q0, *halt*} con q0 stato iniziale. La tabella delle istruzioni della macchina è la seguente:

Stato corrente	Simbolo corrente	Nuovo simbolo	Nuovo stato	Spostamento
q0	1	0	q0	right
q0	0	1	q0	right
q0	*blank*	*blank*	halt	right

Ricordiamo che la configurazione iniziale della macchina include: il contenuto del nastro (di cui al più un sottoinsieme finito di celle può memorizzare un simbolo diverso da *blank*) e la posizione iniziale della testina di lettura-scrittura. Allora:

V *F* Se la macchina viene fatta operare su un nastro in cui tutte le caselle sono *blank*, allora non termina mai (cioé non raggiunge mai lo stato halt)

V *F* Se il contenuto iniziale del nastro è “1001” e la testina di lettura-scrittura viene posizionata sulla cifra più a destra, allora il contenuto del nastro alla fine della computazione è “0110”

V *F* Se il contenuto iniziale del nastro è “0101” e la testina di lettura-scrittura viene posizionata sulla cifra più a sinistra, allora il contenuto del nastro alla fine della computazione è identico al contenuto iniziale.

V *F* Esiste almeno una configurazione iniziale della macchina che fa sì che la MdT non termini mai la computazione, cioè non raggiunga mai lo stato halt.

Esercizio 23 Notazione binaria

Le domande seguenti sono tutte relative alla rappresentazione binaria in complemento a due di numeri interi.

V *F* Se stiamo lavorando in complemento a due con $N = 4$ bit, allora è possibile rappresentare il risultato corretto della somma ($0110_{2C} + 1010_{2C}$).

V *F* Il numero minimo di bit necessari per rappresentare in complemento a due il valore decimale -12 è $N = 6$ bit.

V *F* La rappresentazione decimale di 11001_{2C}, se interpretato in complemento a due con $N = 5$ bit, è -7

V *F* Se lavoriamo in complemento a due con $N = 4$ bit, allora è possibile rappresentare il risultato (esatto) della somma ($0011_{2C} + 0110_{2C}$)

Esercizio 24 Espressioni logiche

Nelle domande seguenti, le variabili *A* e *B* rappresentano valori di input, e *R* rappresenta il risultato di una espressione booleana.

V *F* Consideriamo l'espressione booleana *R* = *A* **AND** (**NOT** *A*) **AND** *B*. Allora esistono dei valori opportuni di *A* e *B* per i quali risulta *R* = 1 (*true*)

V *F* Consideriamo l'espressione booleana *R* = *A* **OR** (**NOT** *A*). Allora esistono dei valori opportuni di *A* per i auali risulta *R* = 1 (*true*)

V *F* Consideriamo l'espressione booleana *R* = *A* **AND** (**NOT** *B*). Allora se *A* e *B* hanno lo stesso valore, il risultato è *R* = 0 (*false*)

V *F* Consideriamo l'espressione booleana *R* = *A* **AND** (**NOT** *B*) (la stessa del punto precedente). Allora per tutte le possibili combinazioni di valori di *A* e *B* risulta sempre *R* = 0 (*false*)

Esercizio 25 Programmazione in C

Consideriamo il seguente frammento di codice in linguaggio C, in cui le varie istruzioni (il cui numero di riga è inidicato all'inizio) sono eseguite nell'ordine mostrato.

```
1. int a[4] = {-2, -3, -4, -5};
2. int* p = &a[1];
3. int* q = p+1;
4. *p = 0;
5. *q = *p;
```

Supponiamo che l'array `a[]` venga memorizzato a partire dall'indirizzo di memoria 12000 (in base 10), e che ogni intero occupi esattamente 4 byte di memoria. Allora:

V *F* Al termine dell'esecuzione della riga 2, il puntatore `p` punta all'indirizzo di memoria 12004

V *F* Al termine dell'esecuzione della riga 3, il puntatore `q` punta all'indirizzo di memoria 12008

V *F* Al termine dell'esecuzione della riga 4, il contenuto dell'array `a[]` è `{0, -3, -4, -5}`

V *F* Al termine dell'esecuzione della riga 5, il contenuto dell'array `a[]` è `{-2, 0, 0, -5}`

Esercizio 26 Grammatiche

Consideriamo la grammatica BNF seguente, dove il simbolo iniziale è <A>, mentre le parole gane, gatto, abbaia, miagola sono simboli terminali.

<A> ::= <B><C>
<B> ::= cane | gatto
<C> ::= abbaia | miagola

Allora:

V *F* La grammatica di cui sopra può generare, tra le altre, la frase “cane abbaia”

V *F* La grammatica di cui sopra può generare, tra le altre, la frase “cane cane”

V *F* La grammatica di cui sopra può generare, tra le altre, la frase “cane miagola”

V *F* La grammatica di cui sopra può generare complessivamente 4 frasi diverse

Esercizio 27 Crittografia

V *F* In un sistema di crittografia simmetrica (detto anche a chiave segreta), le chiavi di cifratura e decifratura sono diverse

V *F* In un sistema di crittografia asimmetrica (detto anche a chiave pubblica), le chiavi di cifratura e decifratura sono diverse

V *F* Supponiamo che in un certo sistema crittografico la chiave di decifratura sia composta da $N = 16$ bit. Allora esistono 2^{16} possibili chiavi di decifratura.

V *F* Consideriamo un cifrario a sostituzione in cui ogni carattere del messaggio “in chiaro” viene sostituito dal carattere immediatamente successivo dell'alfabeto (a → b, b → c, … z → a). Allora il messaggio cifrato “`djbp`” corrisponde al messaggio in chiaro “`ciao`”

Esercizio 28 Grammatiche

Definire una grammatica per rappresentare i numeri interi con segno. La grammatica deve essere in grado di generare, ad esempio: -129, +79, 76, 0. Non deve generare parole tipo 0023, né +0 e -0, ma solo 0 (senza segno).

Esercizio 29 Invarianti

Il frammento di programma seguente calcola il valore minimo in un array $a[]$ di interi di lunghezza $n \geq 1$. Determinare l'invariante di ciclo; non è richiesto di specificare le asserzioni dopo ogni istruzione, è sufficiente l'invariante del ciclo *while*. Assumere che inizialmente valga l'asserzione $\{n \geq 1\}$. Per descrivere le invarianti è possibile ricorrere al linguaggio naturale e/o alla sintassi del linguaggio C per gli operatori aritmetici e logici.

```
int i = 1;
int min = a[0];
while (i<n) {
      if (a[i] < min) {
            min = a[i];
      }
      i = i + 1;
}
```

Esercizio 30 Invarianti

Il frammento di codice seguente *dovrebbe* calcolare il valore della sommatoria $S(n) = 0 + 1 + 2 + \ldots + n$, per un dato intero $n \geq 0$, ma non è corretto. Determinare l'invariante del ciclo, supponendo che inizialmente valga l'asserzione $\{n \geq 0\}$, e verificare che l'invariante non calcola $S(n)$. Per descrivere le invarianti è possibile ricorrere al linguaggio naturale e/o alla sintassi del linguaggio C per gli operatori aritmetici e logici. Correggere quindi il programma in modo da calcolare quanto richeisto.

```
int i = 1, S = 0;
while (i<n) {
      S = S + i;
      i = i + 1;
}
```

Esercizio 31 Invarianti

Scrivere le asserzioni che valgono dopo ciascuna delle istruzioni seguenti, partendo dall'asserzione iniziale { *True* } (cioè inizialmente nulla si sa dei valori di x, y, e z). Verificare che l'invariante descrive in modo formale il risultato intuitivo di questo programma. Per descrivere le invarianti è possibile ricorrere al linguaggio naturale e/o alla sintassi del linguaggio C per gli operatori aritmetici e logici.

```
if ( x > y ) {
      z = x;
} else {
      z = y;
}
```

Esercizio 32 `Invarianti`

Per ognuno dei seguenti frammenti di codice viene data la precondizione (cioè l'asserzione che risulta vera prima dell'esecuzione della prima istruzione). Determinare l'asserzione che vale alla fine; suggerisco di arrivarci per passi, determinando tutte le asserzioni intermedie. Assumere che tutte le variabili abbiano tipo `int`.

1. $\{x < n\}$ `x = x+1; y = 2*x;` { ??? }
2. { *True* } `x=1; y=x+1; x=2*x;` { ??? }
3. $\{i = 0 \wedge i \leq n\}$ `while (i<n) { i=i+1; }` { ??? }

Soluzioni

Esercizio 1

- V
- V
- V: la MdT sposta la testina di lettura/scrittura a destra fino a quando non trova la prima casella blank, che sovrascrive con un 1 prima di terminare. Per ipotesi, inizialmente solo un insieme finito di caselle contiene un 1, quindi la MdT prima o poi troverà una cella contenente *blank* e ne modificherà il contenuto.

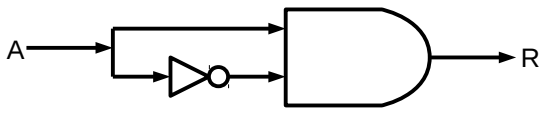
Esercizio 2

Stato corrente	Simbolo corrente	Nuovo simbolo	Nuovo stato	Spostamento
q0	1	*blank*	q0	right
q0	*blank*	*blank*	halt	right

Esercizio 3

Esercizio 4

Il circuito seguente corrisponde all'espressione booleana $R = A$ **AND** (**NOT** A) il cui risultato è sempre *falso*, qualunque sia il valore di A.



Esercizio 5

A	*B*	*R*
0	0	1
0	1	1
1	0	0
1	1	1

Esercizio 6

Osserviamo che *A* ha il primo bit a sinistra posto a zero, quindi rappresenta un numero maggiore o uguale a zero in complemento a due. Invece *B* ha il primo bit a sinistra posto a uno, quindi rappresenta un numero negativo. In particolare, $A = 2 + 4 + 16 + 32 = 54$, mentre $B = 4 + 8 - 128 = -116$. Le risposte sono pertanto:

- V
- F
- V: dalla teoria sappiamo che la somma di un valore positivo e uno negativo in complemento a due non può mai

generare overflow; alternativamente, osserviamo che A + B = -62. che appartiene all'intervallo di valori rappresentabili in complemento a due con 8 bit [-128, 127]
- F
- F

Esercizio 7

Con 5 bit possiamo rappresentare in complemento a due tutti gli interi appartenenti all'intervallo $[-(2^4), 2^4-1] = [-16, 15]$
- F
- V
- V
- V
- F

Esercizio 8

Sono necessari $320 \times 200 \times (5 + 5 + 5) = 960{,}000$ bit = 120,000 B

Esercizio 9

Sappiamo che con N bit possiamo ottenere 2^N configurazioni diverse. La domanda chiede in sostanza di individuare il minimo valore di N per cui $2^N \geq 47$. Tale valore è 6, quindi la risposta corretta è “6 bit”

Esercizio 10

ASCII binario	*ASCII decimale*	*Carattere*
0100 0001	67	C
0100 1001	73	I
0100 0001	65	A
0100 1111	79	O

Esercizio 11

L'auto (e quindi i dati che porta) impiega 32000 m / 16 m/s= 2000 s per percorrere la strada. La latenza è quindi 2000 s. La banda è 32000000000 B / 2000 s = 16000000 B/s, cioè molto meglio di una connessione ADSL! (si noti che la banda delle connessioni ADSL viene quasi sempre presentata in BIT al secondo, mentre nel nostro caso sono 16 milioni di BYTE al secondo).

Esercizio 12

Vedi lucidi

Esercizio 13

Nella crittografia simmetrica si usa la stessa chiave per cifrare e decifrare. Nella crittografia asimmetrica si generano una coppia di chiavi che hanno le proprietà seguenti: (1) deve essere “estremamente difficile” determinare il valore di una delle due chiavi conoscendo il valore dell'altra, e (2) è possibile cifrare con una delle due chiavi, e decifrare esclusivamente con l'altra.

Esercizio 14

- F
- V

- V
- F

Esercizio 15

`INFORMATICA`

Esercizio 16

Una possibile soluzione è la grammatica seguente, in cui l'insieme dei simboli non terminali è {<Rosa>, <NS>, <NO>} con simbolo iniziale <Rosa> e l'insieme dei simboli terminali è {“N”, “S”, “E”, “O”}:

<Rosa> ::= <NS> | <EO> | <NS><EO>
<NS> ::= “N” | “S”
<EO> ::= “E” | “O”

Esercizio 17

- V: entrambe le grammatiche generano parole composte da un numero pari di cifre “0”, oltre alla parola vuota
- F
- F

Esercizio 18

I numeri dispari sono quelli che hanno “1” come ultima cifra a destra quando scritti in notazione binaria. Quindi una possibile soluzione è la grammatica seguente, con simboli non terminali {<NumDispari>, <Cifra>}, simbolo iniziale <NumDispari> e simboli terminali {“0”, “1”}:

<NumDispari> ::= “1” | <Cifra> <NumDispari>
<Cifra> ::= “0” | “1”

Esercizio 19

97

Esercizio 20

- F
- V
- F: la variabile `w` ha valore 47; infatti il puntatore `q` punta al contenuto della variabile `w`, quindi assegnare un valore a `*q` equivale ad assegnare un valore a `w`
- F

Esercizio 21

- F: l'algoritmo di ricerca sequenziale esamina uno per uno tutti i valori della lista, quindi funziona correttamente a prescindere dal fatto che i valori siano ordinati o meno
- F: nel caso migliore il valore cercato è il primo esaminato, quindi nel caso migliore l'algoritmo termina dopo aver esaminato un solo elemento della lista
- V
- V: dato che i valori non sono necessariamente ordinati, non è possibile concludere che un valore non è presente se non dopo averli controllati tutti

Esercizio 22

- F: La MdT di cui sopra, se inizializzata opportunamente, calcola il “complemento a uno” di una stringa binaria (sostituisce tutti gli “1” con “0” e viceversa); in ogni caso la macchina termina sempre, qualunque sia il

contenuto iniziale del nastro e la posizione della testina di lettura/scrittura.

- F: Il contenuto finale del nastro sarà “1000” (viene posto a zero l'ultima cifra a destra, successivamente la testina di lettura/scrittura si sposta a destra e la MdT termina)
- F: Il contenuto finale del nastro sarà “1010”
- F: La MdT termina sempre e comunque.

Esercizio 23

- V: stiamo sommando in complemento a due un valore positivo (6) e uno negativo (-6), per cui sappiamo che il risultato è sempre rappresentabile in complemento a due (nel nostro caso il risultato è 0).
- F: con 6 bit il valore minimo rappresentabile è -32, con 5 bit il valore minimo rappresentabile è -16, e con 4 bit il valore minimo rappresentabile è -8. Quindi bastano $N = 5$ bit.
- V
- F: Se eseguiamo la somma binaria con le regole viste a lezione, otteniamo che il risultato con 4 bit è 1001_{2C}, che in complemento a due rappresenta un valore negativo (-7) mentre gli addendi sono entrambi positivi (3 e 6). Il risultato corretto (9) non è rappresentabile con 4 bit, dato che il valore massimo rappresentabile è 7.

Esercizio 24

- F: Si può scrivere la tabella di verità per constatare che il risultato è sempre false. Possiamo anche osservare che il risultato *R* è l'”and” logico di tre termini, in cui i primi due (*A* e **NOT** *A*) assumeranno sempre valori opposti. Quindi il risultato deve essere sempre *false*.
- V: Si ha $R = 1$ per qualsiasi valore di *A*.
- V: Lo si può verificare sostituendo nell'espressione $A = B = 0$, e $A = B = 1$ e determinando il risultato.
- F: Se $A = 1$ e $B = 0$, il risultato è $R = 1$, quindi l'affermazione è falsa.

Esercizio 25

- V
- V
- F: al termine della riga 4, l'array `a[]` ha contenuto `{-2, 0, -4, -5}`
- V

Esercizio 26

- V
- F
- V
- V

Esercizio 27

- F: nei sistemi a crittografia simmetrica si usa la stessa chiave per cifrare e decifrare
- V
- V
- V

Esercizio 28

Le componenti della grammatica sono le seguenti:
Simboli terminali
“0”, “1”, “2”, … “9”, “+”, “-”
Simboli non terminali
<num>, <segno>, <cifre>, <cifra>, <cifra-nz>
Regole di produzione

<num> ::= “0” | <segno> <cifra-nz> <cifre>
<segno> ::= e | “+” | “-”
<cifre> ::= e | <cifra> <cifre>
<cifra> ::= “0” | “1” | “2” | ... | “9”
<cifra-nz> ::= “1” | “2” | ... | “9”

Simbolo iniziale
<num>

Esercizio 29

```
/* n >= 1 */
int i = 1;
int min = a[0];
/* min è il minimo in a[0..i-1] && i<=n */
while (i<n) {
      /* min è il minimo in a[0..i-1] && i<n */
      if (a[i] < min) {
            min = a[i];
      }
      i = i+1;
      /* min è il minimo in a[0..i-1] && i<=n */
}
/* min è il minimo in a[0..i-1] && i==n */
```

Esercizio 30

```
/* n >= 0 */
int i = 0, S = 0;
/* S = 0 + 1 + … + (i-1) && n>=0 */
while (i<n) {
      /* S = 0 + 1 + … + (i-1) && i<n */
      S = S + i;
      i = i + 1;
      /* S = 0 + 1 + … + (i-1) && i<=n */
}
/* S = 0 + 1 + … + (i-1) && i==n */
```

Quindi il programma calcola 0 + 1 + 2 + … + (*n* - 1); per calcolare *S*(*n*) è necessario modificare la condizione del ciclo while in (`i <= n`)

Esercizio 31

```
/* True */
if ( x > y ) {
      /* x > y */
      z = x;
      /* x > y && z == x */
} else {
      /* x <= y */
      z = y;
      /* x <= y && z == y */
}
/* (x>y && z=x) || (x<=y && z=y) */
```

Si può notare come l'invariante finale equivalga a $z = \max(x, y)$

Esercizio 32

1. {$x < n$} `x = x+1` {$x \leq n$}`; y = 2*x;` { $x \leq n \land y \leq 2n$ }
2. { *True* } `x=1` { x=1 }`; y=x+1` { x=1 $\land$ y=2 }`; x=2*x;` { x=2 $\land$ y=2 }
3. Osservando il codice si può notare come al termine del ciclo while si abbia sempre che $i = n$; prima del corpo del ciclo si ha sempre che $i < n$, e al termine di ogni esecuzione del corpo del ciclo si ha che $i \leq n$. Proviamo quindi ad utilizzare $i \leq n$ come invariante. Utilizzando la notazione vista a lezione, abbiamo che I = $i \leq n$ e C = {$i < n$} (C rappresenta la condizione del ciclo).
   {$i = 0 \land i \leq n$} `while (i<n) {` {I $\land$ C} `i=i+1;` {I} `}` { I $\land$ $\neg$C }
   sostituendo si ha
   {$i = 0 \land i \leq n$} `while (i<n) {` {$i \leq n \land i < n$} `i=i+1;` {$i \leq n$} `}` { $i \leq n \land i \geq n$ }
   che può essere semplificato in
   {$i = 0 \land i \leq n$} `while (i<n) {` {$i < n$} `i=i+1;` {$i \leq n$} `}` { $i = n$ }