UNIVERSITÀ DI PISA
Corso di Laurea in Ingegneria Informatica
c/o Dipartimento di Ingegneria dell'Informazione: Elettronica, Informatica, Telecomunicazioni

Esami di Ingegneria del Software
e di Ingegneria dei Sistemi Software

Appello del 22 aprile 2005

Nome e cognome:
Matricola:
Esame: **Ing. del Software** ☐ **Ing. dei Sistemi Software** ☐

Il punteggio in trentesimi associato a ciascuna domanda è indicato fra parentesi. Ove si trovino coppie di domande con lo stesso numero, rispondere solo a quella corrispondente all'esame che si vuole sostenere: la domanda per l'esame di Ingegneria del Software (vecchio ordinamento) è identificata dalla sigla [**V**], quella per l'esame di Ingegneria dei Sistemi Software (nuovo ordinamento) dalla sigla [**N**].

Scegliere una risposta per ciascuna domanda:

1 Una classe astratta (1)
Non è implementata. ☐
È parzialmente implementata. ☒
Non è implementabile. ☐

2 Negli Statechart un'azione è (1)
un comportamento non interrompibile. ☒
una transizione. ☐
un comportamento interrompibile. ☐

3 Il debugging è la ricerca e rimozione (1)
degli errori. ☐
dei difetti. ☒
dei malfunzionamenti. ☐

4 Il significato delle formule è dato (1)
dalla sintassi del linguaggio. ☐
dalla semantica del linguaggio. ☒
dalle regole d'inferenza. ☐

5 Il test di integrazione serve collaudare (1)
il sistema completo. ☐
i singoli moduli. ☐
l'interfacciamento fra i moduli. ☒

Rispondere alle domande, usando solo lo spazio disponibile:

6 Nelle reti di Petri, quando si dice che una transizione è viva? (3)

Per ogni marcatura M raggiungibile da M_0, esiste una sequenza di scatti che a partire da M abilita la transizione.

7 Definire il criterio di copertura delle decisioni. (3)

Ogni arco del grafo di controllo deve essere percorso almeno una volta.

8 Scrivere la tabella di verità dell'operatore di coimplicazione ($\Leftrightarrow$). (3)

A	B	$A \Leftrightarrow B$
T	**T**	*T*
T	**F**	*F*
F	**T**	*F*
F	**F**	*T*

9 Che cos'è una classe? (3)

La descrizione di un insieme di oggetti con struttura e comportamento simili. (Oppure: rappresentazione di un concetto pertinente al sistema, "reale" o software, che viene specificato).

10 Che cos'è un prototipo? (3)

Una versione approssimata, parziale, ma funzionante, dell'applicazione che viene sviluppata.

11 Con riferimento alla Fig. 1, rispondere alle domande. (4)

	V	F
La classe **Key** ha solo l'op. less()	☐	☒
La classe **Map** implementa l'op. less()	☐	☒
L'op. next() non dipende da **Key** e **Value**	☒	☐
Gli oggetti **Pair** sono ordinati in base a second	☐	☒

12 Supponendo che siano disponibili in C++: (6)

(a) l'implementazione delle classi e interfacce dell'esercizio precedente (le relative dichiarazioni siano in `map.h`); e
(b) l'implementazione della classe `string` dotata dell'operatore `operator<(const string& s)` per l'ordinamento alfabetico (dichiarazioni in `string.h`);
(1) realizzare due classi **Name** e **Address** da usare per memorizzare coppie $<$*nome,indirizzo*$>$ in una struttura **Map** (tali classi devono avere anche un'operazione che restituisca, risp., il nome o l'indirizzo sotto forma di `string`);
(2) implementare la funzione globale `print(const Map& m)` che scrive sull'uscita standard gli elementi di una struttura **Map**.

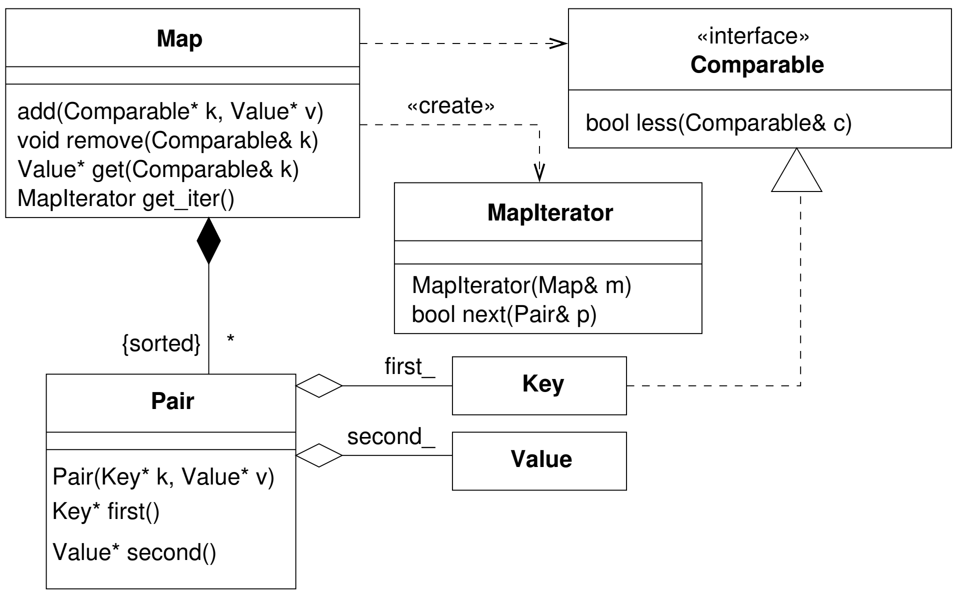

Figura 1: Domanda 11.

```
#include <iostream>
#include "map.h"
#include "string.h"

using namespace std;

class Name : public Key {
    string s_;
public:
    Name(const string& s) : s_(s) {};
    bool less(Comparable& c);
    string str() const { return s_; };
};

class Address : public Value {
    string s_;
public:
    Address(string s) : s_(s) {};
    string str() const { return s_; };
};

bool
Name::
less(Comparable& c)
{
    return (s_ < dynamic_cast<Name*>(&c)->str());

    // OPPURE:
    // return (s_ < ((Name*)&c)->str());
}

void
print(const Map& m)
{
    MapIterator i = m.get_iter();
    Pair p;
    while(i.next(p)) {
        Name* np = dynamic_cast<Name*>(p.first());
        Address* ap = dynamic_cast<Address*>(p.second());

    // OPPURE:
        Name* np = (Name*)p.first();
        Address* ap = (Address*)p.second();

        string ns = np->str();
        string as = ap->str();
        cout << ns << endl;
        cout << as << endl;
    }
}
```


Figura 2: Domanda 12, soluzione.