FONDAMENTI DI PROGRAMMAZIONE B

Tempo a disposizione: 20 minuti

Nome ..... Cognome ..... Matricola .....

Per accedere alla prova di programmazione è necessario rispondere correttamente ad almeno il 70% delle domande

1. [C++] Se una classe non definisce esplicitamente il distruttore

- [ ] *a* viene generato un errore a tempo di compilazione
- [ ] *b* viene generato un errore a tempo di esecuzione
- [x] *c* la classe ha un distruttore di default
- [ ] *d* non è possibile istanziare oggetti di quella classe
- [ ] *e* nessuna delle precedenti

2. [C++] Si indichi quale dei seguenti è l'esempio più appropriato di costruttore di copia per una classe `C`.

- [ ] *a* `C(C *other) {...}`
- [ ] *b* `C(C other) {...}`
- [x] *c* `C(const C &other) const {...}`
- [ ] *d* `C(const C &other) {...}`

3. [C++] Si considerino le classi `A`, `B`, `C` e `D`. La classe `B` è derivata da `A`, mentre le classi `C` e `D` sono derivate da `B`. Il seguente metodo `foo` della classe `B`

```
void shift(B obj) {...}
```

può accettare come argomenti oggetti

- [ ] *a* di tipo `A`, `B` e `Object`
- [ ] *b* di tipo `B` ma non di tipo `C` e `D`
- [ ] *c* di tipo `A` e `B`
- [x] *d* di tipo `B`, `C` e `D`

4. [C++] Una classe `C` ha sempre a disposizione almeno un costruttore [x] T [ ] F

5. [C++] Data una classe `C`, la dichiarazione `C c;` inizializza la variabile `c` a `nullptr`. [ ] T [x] F

6. [**Java**] Si consideri la seguente dichiarazione di attributo all'interno di una classe `C`:

```
private static int x;
```

Si indichi la risposta corretta.

- [ ] *a* è un attributo d'istanza con visibilità privata e non modificabile (dopo la sua inizializzazione nel costruttore)
- [ ] *b* è un attributo di classe con visibilità privata e non modificabile (dopo la sua inizializzazione nel costruttore)
- [x] *c* è un attributo di classe con visibilità privata e modificabile
- [ ] *d* è un attributo d'istanza con visibilità privata e modificabile
- [ ] *e* è un attributo d'istanza con visibilità di package e modificabile
- [ ] *f* nessuna delle precedenti

7. [**Java**] Si considerino le classi `A`, `B` e `C`. Le classi `C` e `B` estendono `A`. La classe `A` definisce un metodo `foo` che viene sovrascritto sia dalla classe `B` che dalla classe `C`. Si consideri il seguente frammento di codice.

```
B obj = new C();
((A) obj).foo();
```

- [ ] *a* viene sollevata una `ClassCastException` a tempo di esecuzione
- [ ] *b* viene ritornato un errore a tempo di compilazione
- [ ] *c* viene invocato il metodo `foo` definito nella classe `A`
- [ ] *d* viene invocato il metodo `foo` definito nella classe `B`
- [x] *e* viene invocato il metodo `foo` definito nella classe `C`
- [ ] *f* nessuna delle precedenti

8. [**Java**] Data la classe `A`, si consideri il seguente frammento di codice

```
A x = new A();
A y = x;
```

Dopo l'esecuzone del frammento di codice riportato sopra

- [ ] *a* `y` fa riferimento ad un oggetto che è una copia leggera (*shallow copy*) dell'oggetto riferito da `x`
- [ ] *b* `y` fa riferimento ad un oggetto che è una copia profonda (*deep copy*) dell'oggetto riferito da `x`
- [x] *c* `x` e `y` fanno riferimento allo stesso oggetto
- [ ] *d* viene sollevata un'eccezione a tempo d'esecuzione
- [ ] *e* nessuna delle precedenti

9. [**Java**] Un parametro di tipo di una classe generica (e.g., `Stack`⟨`T`⟩) può essere specializzato con tipi primitivi. [ ] T [x] F

10. [**Java**] Si consideri una classe `C`. L'istruzione `String s;` è equivalente all'istruzione `String s = "";` (stringa vuota) [ ] T [x] F