C++

Nell'esercizio di C++ è prevista la realizzazione di un programma di simulazione di una corsa a tempo tra animali.

L'esercizio è composto da una parte obbligatoria e una facoltativa.

Il codice deve essere correttamente compilabile e accompagnato dal rispettivo Makefile funzionante (se il Makefile è assente saranno decurtati dei punti). Non saranno accettati codici non compilabili (abbiate cura di commentare la parte di codice che non riuscite sviluppare. In caso di indecisione sul voto tale parte sarà comunque parzialmente valutata).

Il codice deve rispondere correttamente alle richieste previste nel main. In sede di valutazione il codice sarà comunque testato per verificarne l'affidabilità.

Parte Obbligatoria

Il programma di simulazione realizza una gara a tempo tra corridori (animali) diversi e di razze diverse. Vince la gara il corridore che nel tempo fissato percorre lo spazio maggiore.

Ogni razza di animale (cane, cavallo, coniglio …) è caratterizzata da una curva caratteristica velocità verso tempo v(t) parametrica.
Diversi animali della stessa razza sono differenziati da differenti parametri della curva caratteristica.

Di seguito sono descritte le funzionalità richieste per le varie classi costituenti il sistema corredate da suggerimenti e parti di codice.

Classe *Animale*

La classe *Animale* rappresenta la classe base per la descrizione del corridore da cui gli animali "concreti" dovranno ereditare.
L'interfaccia della classe è descritta dalla seguente dichiarazione:

```
class Animale {
public:
   Animale(char * name);
   virtual double getSpeed(double time) = 0;
   char * getName();
   void setPosition(double position);
   double getPosition() const;
private:
   char name[80];
   double position;
};
```

Il metodo puramente virtuale *getSpeed(t)* sarà implementato nelle sottoclassi specifiche e ritorna la

velocità in funzione del tempo.
La classe *Animale* contiene le variabili *name* e *position* che memorizzano rispettivamente il nome e la posizione corrente del corridore.

Classe *Reporter*

La classe *Reporter* mediante il metodo *report(Animale*)* scrive sullo schermo il nome e la posizione dell'animale dato.

```
class Reporter {
public:
   Reporter();
   void report ( Animale * );
};
```

Classe *Gara*

La classe gara descritta dalla seguente dichiarazione :

```
class Gara {
public:
   Gara ( int duration, Reporter * r );
   void add(Animale*);
   void start();
   void classifica();

private:

   void sortAnimali();

   Animale* animali[100];
   Reporter * theReporter;
   int size;
   int duration;
};
```

mantiene memoria degli appartenenti alla gara e ne gestisce lo svolgimento.

Il metodo *add(Animale*)* permette di aggiungere un nuovo animale alla gara.

La chiamata del metodo *start()* implementa la gara: la durata complessiva in secondi *duration* della gara viene suddivisa in unità discrete di 1 secondo. Per ogni passo temporale viene richiesto ad ogni animale la propria velocità a quel tempo, viene calcolata la nuova posizione e aggiornata nell'animale stesso (mediante il metodo *setPosition*).
Per ogni passo temporale viene stampata a schermo la "classifica" parziale mediate l'oggetto Reporter passato alla gara in fase di inizializzazione.

Il metodo *classifica()* al termine della gara permette di stampare, usando il Reporter, la classifica finale

della gara in ordine di arrivo.
Per ordinare il vettore di Animali prima della stampa della classifica implementare il metodo sortAnimali() che ordina gli animali in base allo spazio percorso.

Classe *Cavallo*

La classe Cavallo eredita dalla classe madre Animale ed implementa la funzione *getSpeed(double t)* con la seguente funzione:

$$v(t) = vmax * (1 – exp (-t / exppar))$$

Implementare il costruttore della classe Cavallo che accetti, oltre al nome, i due parametri della funzione:

```
Cavallo(char * name, double vmax, double exppar );
```

N.B. : la funzione di sistema *exp* ha il seguente prototipo:

```
double exp(double x);
```

ed è definita nel file *math.h*

Classe Cane

La classe Cane eredita dalla classe madre Animale ed implementa la funzione *getSpeed(double t)* con la seguente funzione:

$$v(t) = vmax$$

Implementare il costruttore della classe Cane che accetti, oltre al nome, il parametro della funzione:

```
Cane(char * name, double vmax);
```

Main

- Creare una gara con 4 animali, 2 cavalli e 2 cani con parametri diversi
- Eseguire il metodo *start* di gara
- Stampare la classifica della gara invocando il metodo *classifica*

Parte facoltativa

a) Introdurre una variazione casuale del 20% della velocità ritornata da *getSpeed*. Utilizzare la funzione *int rand(void)* definita nell'header di sistema *stdlib.h*

b) Implementare l'operatore > tra due animali che determina una relazione d'ordine in base allo spazio percorso. Utilizzare l'operatore nel metodo *sortAnimali()* di Gara.
Il prototipo dell'operatore è:

```
bool operator >(const Animale & a1, const Animale & a2);
```

c) implementare la classe puramente virtuale *VReporter*. Modificare la classe *Reporter* in modo che erediti da *VReporter*. Modificare di conseguenza la classe *Gara.*

d) implementare un secondo tipo di *VReporter*, *FileReporter*, che stampa gli animali su file invece che a schermo

e) utilizzando argc e argv nel main implementare la possibilità di scegliere quale tipo di *VReporter* utilizzare

f) implementare un nuovo animale concreto mediante la classe *Coniglio* con una propria funzione parametrica della velocità.