Programmazione 1

Esami del 2020

Stefano Gualandi

January 8, 2021

Premessa

Questo documento presenta la raccolta dei testi di esame degli appelli del corso di Programmazione 1 tenuto nel 2020 agli studenti del primo anno del corso di laurea triennale di matematica, presso l'Università di Pavia.

I files dati citati nei testi d'esame sono disponibili al sito:

`http://www-dimat.unipv.it/gualandi/programming/exams/Raccolta2020.zip`

IMPORTANTE: I testi di questi esami possono essere utilizzati per svolgere degli esercizi. In nessun modo si deve intendere che questi temi di esame sono indicativi di quanto verrà chiesto agli appelli futuri. Quanto verrà richiesto agli esami futuri, rifletterà direttamente il programma svolto a lezione nell'anno accademico corrispondente.

Programmazione 1 **Esame del 27 gennaio 2020**

Cognome: **Nome:** **Matricola:** **Postazione:**

Prima Parte

Nella prima parte dell'esame dovete usare esclusivamente le funzioni ricorsive e le `pairlists`, come studiate nella prima parte del corso. Dovete importare la libreria `pairlists` che viene fornita nella vostra cartella d'esame. Dovete svolgere i seguenti esercizi, completando il file `soluzione_template.py` e **utilizzando le funzioni di test che trovate nel blocco `main` in coda al file**.

1. **(Punti 3)** Scrivere una funzione che prende in input una lista `As` di coefficienti di un polinomio di ordine n, e restituisce due funzioni, la prima funzione permette di calcolare in un dato punto la derivata prima del polinomio, la seconda funzione permette di calcolare la derivata seconda del polinomio. Ad esempio, dato il polinomio:

   $$3x^4 + 5x^3 + x - 2$$

   i cui coefficienti sono $a_4 = 3, a_3 = 5, a_2 = 0, a_1 = 1, a_0 = -2$, il polinomio viene rappresentato dalla lista `As = [3,5,0,1,-2]`. Si deve scrivere una funzione `F(As)` che può essere usata nel modo seguente:

   ```
   G, H = F(As)
   print('Derivata prima in x=2:', G(2))
   print('Derivata seconda in x=2:', H(2))
   ```

2. **(Punti 4)** Scrivere una funzione `Zip(As, Bs)` che prende in input due pairlists della stessa lunghezza, e restituisce una lista di coppie di elementi, in cui l'ennesimo elemento di `As` é accoppiato con l'ennesimo elemento di `Bs`.

   Modificare la funzione `Zip(As, Bs)` dell'esercizio precedente in modo tale che quando le due liste non hanno la stessa lunghezza, viene sollevata un'eccezione di tipo `Runtime`.

3. **(Punti 6)** Scrivere una funzione che implementa l'algoritmo di *merge-sort* utilizzando le `pairslist`. La funzione `MergeSort(Ls, Compare)` da implementare, prende in input una lista di elementi (lista di tipo `pairslist`), e una predicato di confronto `Compare(x, y)` tra due elementi della lista, e restituisce in output una nuova lista ordinata in base alla funzione di confronto `Compare(x, y)` data in input.

Seconda Parte

Nella seconda parte dell'esame potete usare qualsiasi libreria di Python.

Si consideri un file di testo, chiamato `cells25.csv`, che contiene i risultati di un'analisi medica di un paziente malato di leucemia. Per ogni riga del file, tranne la prima che contiene la descrizione di ogni campo, vengono riportate le seguenti informazioni:

```
Patient:        nome in codice del paziente
CellName:       nome della cellula
Label:          stato della cellula "0=normal" oppure "1=malignant"
Gene1:          segue una lista di "gene expression": nome gene -> quantità espressa
Gene2:          si ha un numero intero per ogni gene (per ogni colonna)
...             ...
Gene27836:      ci sono 27836 geni, ma molti non sono espressi (=0)
```

4. **(Punti 3)** Si scriva una classe `Cell` con quattro attributi: `Patient`, `Cell`, `Label` e `Genes`. I primi due attributi sono delle stringhe. Il terzo è un valore logico booleano, pari a `True` se la cellula è maligna, e a `False` altrimenti. Il quarto attributo è una lista di numeri interi, un numero per ogni gene (colonna) letta dal file. La classe deve contenere almeno un metodo costruttore `__init__(self, row)`, che data una riga letta dal fila, inizializza un oggetto corrispondente di tipo `Cell`.

5. **(Punti 1)** Si implementi il metodo `__str__(self)`, che restituisce una stringa contenente i primi tre attributi, nel formato seguente:

   `"Patient:  GSM3587923_AML1012-D0, Cell:  AML1012-D0_GGAACCCACTAN, Label:  1"`

6. **(Punti 3)** Si implementi il metodo `__lt__(self, other)`, che restituisce True se la cellula `self` è minore della cellula `other`. Una cellula è minore di un'altra, se la somma totale dei valori espressi in un gene è inferiore al totale dei geni espressi nell'altra cellula.

7. **(Punti 3)** Si scriva una funzione `ParseCells(filename)`, che prende in input il nome di un file, e restituisce una lista di oggetti di tipo `Cell`, un oggetto per ogni riga del file (a parte la prima riga di intestazione).

8. **(Punti 3)** Si scriva una funzione `TopCells(Ls, n=5)`, che prende in input una lista di oggetti come restituiti dalla funzione `ParseCells`, e che restituisce le prime n cellule con la somma totale di geni espressi più alta.

9. **(Punti 6)** Si scriva una funzione `TopGenes(Ls)`, che prende in input una lista di oggetti come restituiti dalla funzione `ParseCells`, e che restituisce sei indici (ovvero sei numeri interi): (i) i primi tre indici corrispondono ai tre geni con il valore maggiore di espressione dei geni tra le cellule sane; (ii) gli altri tre indici corrispondono ai tre geni con il valore maggiore di espressione dei geni tra le cellule malate.

Programmazione 1 **Esame del 12 febbraio 2020**

Cognome: **Nome:** **Matricola:** **Postazione:**

Prima Parte

Nella prima parte dell'esame dovete usare esclusivamente le funzioni ricorsive e le `pairlists`, come studiate nella prima parte del corso. Dovete importare la libreria `pairlists` che viene fornita nella vostra cartella d'esame. Dovete svolgere i seguenti esercizi, completando il file `sol_template.py` e **utilizzando le funzioni di test che trovate nel blocco `main` in coda al file**.

1. **(Punti 3)** Una funzione f è definita dalla regola seguente:

$$f(n) = \begin{cases} n & \text{if } n < 3 \\ f(n-1) + 2\,f(n-2) + 3\,f(n-3) & \text{if } n \geq 3 \end{cases}$$

   (a) Si scriva una funzione che calcoli f usando un **processo ricorsivo**.

   (b) Si scriva una funzione che calcoli f usando un **processo iterativo**.

2. **(Punti 4)** Scrivere una funzione `Integrale(F, a, b, dx)` che calcoli l'integrale definito di una funzione f tra i limiti a e b, che può essere approssimato numericamente usando la formula:

$$\int_a^b f(x)\,dx = [f(a + dx/2) + f(a + dx/2 + dx) + f(a + dx/2 + 2dx) + ...]dx$$

   per valori piccoli di dx.

3. **(Punti 4)** Scrivere una funzione che implementa l'algoritmo *quick-sort* utilizzando le `pairslist`. Implementate la funzione `QuickSort(Ls, Compare)` che prende in input una lista di numeri interi (lista di tipo `pairslist`), un predicato di confronto `Compare(x, y)` tra due numeri interi, e restituisce una nuova lista ordinata in base alla funzione `Compare`.

Seconda Parte

Nella seconda parte dell'esame potete usare qualsiasi libreria di Python.

Si consideri un file di testo, chiamato `koby.csv`, che contiene i risultati di (quasi) tutti i tiri su azione provati da Koby Bryant (1978-2020) durante la sua carriera sportiva nel NBA. Per ogni riga del file, tranne la prima che contiene la descrizione di ogni campo, vengono riportate le seguenti informazioni:

```
combined_shot_type: tipo di azione
game_id:            numero intero, identificativo unico della partita
minutes_remaining:  numero di minuti alla fine del tempo
period:             numero del tempo (se maggiore di 5, sono supplementari)
playoffs:           pari a 1 se partita di play-off
season:             stagione del campionato
seconds_remaining:  numero di secondi alla fine del tempo
shot_distance:      distanza di tiro
shot_made_flag:     numero intero, pari a 1 se il tiro è stato realizzato, 0 altrimenti
shot_type:          numero intero, tipo di tiro realizzato (2 o 3 punti)
shot_zone_area:     zona di tiro
game_date:          stringa, data nel formato 'gg/mm/anno'
opponent:           sigla squadra avversaria
```

4. **(Punti 4)** Si scriva una classe `Shot` con quattro attributi: `game_date`, `shot_made_flag`, `shot_type` e `game_id`. Il primo attributo è una stringa, gli altri sono dei numeri interi. La classe deve contenere il metodo costruttore `__init__(self, row)`, che data una riga letta dal fila, inizializza un oggetto corrispondente di tipo `Shot`. La classe deve contenere anche il metodo `__str__(self)`, che restituisce una stringa contenente i primi tre attributi, nel formato seguente:

   `"Game: 20000012, Date: 31/10/2000, Made: 1, Type: 2"`

5. **(Punti 3)** Si scriva una funzione `ParseFile(filename)`, che prende in input il nome di un file, e restituisce una lista di oggetti di tipo `Shot`, un oggetto per ogni riga del file (a parte la prima riga di intestazione). Il carattere di separazione tra i caratteri è il ";" (punto e virgola).

6. **(Punti 2)** Si scriva una funzione `MakeDict(Ls)`, che prende in input una lista di oggetti come restituiti dalla funzione `ParseFile`, e che restituisce un dizionario con una chiave per ogni partita giocata, a cui sono associati come valore i punti realizzati in quella partita.

7. **(Punti 3)** Si scriva una funzione `Histogram(Ds)`, che prende in input in dizionario come restituito dalla funzione `MakeDict(Ls)`, e restituisce una coppia di valori. Il primo elemento della coppia indica il numero massimo di punti realizzati su azione in una singola partita; il secondo elemento della coppia è una lista di 10 elementi: il primo elemento della lista contiene il numero di partite in cui Koby Bryant ha realizzato da 0 a 9 punti, il secondo elemento della lista, da 10 a 19 punti, il terzo da 20 a 29, ... l'ultimo elemento della lista, da 90 a 99 punti.

8. **(Punti 5)** Si scriva una funzione `MorePoints(Ls, n)`, che prende in input la lista di oggetti letta dal file `Ls`, e un numero intero `n`, e restituisce i `game_id` delle prime n partite con il maggior scarto positivo tra punti realizzati e tiri provati.

9. **(Punti 4+1)** Si scriva una funzione `Plots(Ls, year)`, che prende in input la lista di oggetti letta dal file `Ls`, e mostra, per tutto le gare dell'anno `year` indicato in input, il plot di due funzioni lineari a tratti: la prima mostra il numero di tiri fatti in funzione del giorno di gara, la seconda il numero di punti realizzati in funzione del giorno di gara.

   Verrà valutato in modo positivo il riuscir a fare il plot della stagione invece dell'anno solare. In quel caso, l'anno di input indica l'inizio della stagione. Per esempio, se `year` vale 1999, allora si deve fare il plot delle partite giocate nella stagione 1999/2000 (tra settembre e giugno).

Programmazione 1 **Esame del 24 giugno 2020**

Cognome: **Nome:** **Matricola:** **Postazione:**

Prima Parte

Nella prima parte dell'esame dovete usare esclusivamente le funzioni ricorsive e le `pairlists`, come studiate nella prima parte del corso. Dovete importare la libreria `pairlists` che viene fornita nella vostra cartella d'esame. Dovete svolgere i seguenti esercizi, completando il file `soluzione_template.py` e **utilizzando le funzioni di test che trovate nel blocco `main` in coda al file**.

1. **(Punti 3)** Scrivere una funzione `Norma(x, t)` che prende in input una lista di numeri `x` (ovvero un vettore di $\mathbb{R}^n$) e un intero t e restituisce la norma del vettore. Se $t = 0$ restituisce la norma infinito, ovvero $\max_i |x_i|$; se $t = 1$ calcola la norma 1, ovvero $\sum_i |x_i|$; se $t = 2$ la norma 2, ovvero $\sqrt{\sum_i x_i^2}$.

2. **(Punti 4)** Scrivere una funzione `Integrale(F, a, b, dx)` che calcoli l'integrale definito di una funzione f tra i limiti a e b, che può essere approssimato numericamente usando la formula:

$$\int_a^b f(x)\,dx = [f(a + dx/2) + f(a + dx/2 + dx) + f(a + dx/2 + 2dx) + ...]dx$$

   per valori piccoli di dx.

3. **(Punti 6)** Scrivere una funzione che implementa l'algoritmo di *merge-sort* utilizzando le `pairslist`. La funzione `MergeSort(Ls, Compare)` da implementare, prende in input una lista di elementi (lista di tipo `pairslist`), e una predicato di confronto `Compare(x, y)` tra due elementi della lista, e restituisce in output una nuova lista ordinata in base alla funzione di confronto `Compare(x, y)` data in input.

Seconda Parte

Nella seconda parte dell'esame potete usare qualsiasi libreria di Python.

Si consideri un file di testo, chiamato `dpc-covid19-ita-regioni-latest.csv`, che contiene i dati della protezione civile riguardo l'epidemia in corso, rilasciati il 21 giugno 2020 su GitHub. Per ogni riga del file, tranne la prima che contiene la descrizione di ogni campo, vengono riportate le seguenti informazioni:

```
 0, data: data a cui si riferiscono le informazioni
 1, stato: codice dello stato
 2, codice_regione: codice numerico per la regione
 3, denominazione_regione: nome della regione
 4, lat: latitudine
 5, long: longitudine
 6, ricoverati_con_sintomi: numero di pazienti ricoverati con sintomi
 7, terapia_intensiva: numero di pazienti in terapia intensiva
 8, totale_ospedalizzati: numero di pazienti ospedalizzati
 9, isolamento_domiciliare: numero di pazienti in isolamento domiciliare
10, totale_positivi: numero di pazienti positivi in totale dall'inizio dell'epidemia
11, variazione_totale_positivi: numero di pazienti positivi rispetto al giorno precedente
12, nuovi_positivi: numero di nuovi pazienti positivi per la data di riferimento
13, dimessi_guariti: numero di pazienti guariti
14, deceduti: numero di pazienti deceduti
```

```
15, totale_casi: totale dei casi registrati
16, tamponi: totale dei tamponi eseguiti
17, casi_testati: numero totale dei casi testati
18, note_it: eventuali note in italiano
19, note_en: eventuali note in inglese
```

4. **(Punti 2)** Si scriva una classe `Region` con i seguenti attributi: `Data`, `Nome`, `TerapiaIntensiva`, `NuoviPositivi`, `Deceduti`, `Totale`, e `Tamponi`. Tranne il nome che è una stringa, tutti gli altri attributi sono dei numeri interi. La classe deve contenere almeno un metodo costruttore `__init__(self, row)`, che data una riga letta dal fila, inizializza un oggetto corrispondente di tipo `Region`.

5. **(Punti 1)** Si implementi il metodo `__str__(self)`, che restituisce una stringa contenente i primi tre attributi, nel formato seguente:

   `"Regione: Lombardia, Totale: X, Deceduti: X, Tamponi: X"`

6. **(Punti 2)** Si implementi il metodo `__lt__(self, other)`, che restituisce True se la regione `self` ha avuto un rapporto tra il numero di morti e il totale dei pazienti infetti inferiore allo stesso rapporto dell'altra regione codificata nell'oggetto `other`.

7. **(Punti 3)** Si scriva una funzione `Parse(filename)`, che prende in input il nome di un file, e restituisce una lista di oggetti di tipo `Region`, un oggetto per ogni riga del file (a parte la prima riga di intestazione).

8. **(Punti 3)** Si scriva una funzione `TopRegion(Ls, n=5)`, che prende in input una lista di oggetti di tipo `Region` come restituiti dalla funzione `Parse`, e che restituisce le prime n regioni con il maggior rapporto tra numeri di morti e pazienti infetti.

9. **(Punti 3)** Si scriva una funzione `ParseFiles(names)` che prende in input una lista di nomi di files che devono essere letti. La funzione per ogni nome della lista, deve richiamare la funzione `Parse`, leggere il contenuto del file ottenendo una lista di oggetti di tipo `Region`, e deve restituire alla fine la lista complessiva di oggetti letti dai file di input.

   Nella funzione di test, si chiede di leggere cinque file, dello stesso tipo del file `.csv` descritto sopra, ma che si riferiscono a date diverse.

10. **(Punti 5)** Si scrive una funzione `ComputeAverage(Ls)`, che prende in input una lista di oggetti di tipo `Region`, letti per esempio con la funzione `ParseFiles(names)`, e calcola, per ogni regione, la media aritmetica giornaliera dei nuovi casi registrati. La funzione deve restituire un dizionario con una chiave per ogni regione, a cui corrisponde come valore la media giornaliera calcolata di nuovi casi.

Programmazione 1 **Esame del 14 luglio 2020**

Cognome: **Nome:** **Matricola:** **Postazione:**

Prima Parte

Nella prima parte dell'esame dovete usare esclusivamente le funzioni ricorsive e le `pairlists`, come studiate nella prima parte del corso. Dovete importare la libreria `pairlists` che viene fornita nella vostra cartella d'esame. Dovete svolgere i seguenti esercizi, completando il file `soluzione_template.py` e **utilizzando le funzioni di test che trovate nel blocco `main` in coda al file**.

1. **(Punti 3)** Scrivere una funzione `Zip(As, Bs)` che prende in input due pairlists della stessa lunghezza, e restituisce una lista di coppie di elementi, in cui l'ennesimo elemento di `As` é accoppiato con l'ennesimo elemento di `Bs`.

2. **(Punti 7)** Utilizzando le pairslist, si scrivano le due funzioni high-order seguenti:

   (a) `Map(F, Ls)` che prende in input una funzione `F` e una lista `Ls`, e restituisce una nuova lista che risulta dall'aver applicato la funzione `F` ad ogni elemento di `Ls`.

   (b) `Filter(P, Ls)` che prende in input un predicato `P` e una lista `Ls`, e restituisce una nuova lista contenente solo gli elementi di `Ls` per cui il predicato `P` è soddisfatto (i.e., restituisce `True`).

   Si controlli il funzionamento delle due funzioni usando le funzioni di test date nel `__main__`. Dopo aver verificato che sono corrette, riscrivere le due funzioni in termini di una funzione `Reduce` che dovete implementare voi. Le due nuove funzione sono chiamate `MapReduce` e `FilterReduce`.

Seconda Parte

Nella seconda parte dell'esame potete usare qualsiasi libreria di Python.

Si consideri un file di testo, chiamato `digits.csv`, contenente un sotto campione di immagini digitalizzate di alcune cifre, memorizzate come un **vettore** di $28 \times 28 = 784$ pixels. Il file contiene, in ogni riga, un numero che indica la cifra memorizzata, seguita da un vettore di 784 elementi. Ogni elemento è un numero compreso tra 0 e 255 che indica l'intensità di grigio del corrispondente pixel. Per esempio, le prime tre righe del file potrebbero essere (per ogni riga, dei 784 pixels, riportiamo solo i primi sei e gli ultimi due):

```
5,255,248,248,230,191,168,...,214,198
5,240,250,252,250,248,254,...,207,203
6,255,250,255,252,250,254,...,232,241
```

4. **(Punti 2)** Si scriva un predicato `IsPosInteger(w)` che prende in input una stringa `w` e restituisce `True` se e solo se la stringa `w` contiene solo caratteri numerici, e può essere convertito in un numero intero positivo. Ad esempio, `IsPosInteger('17')` restituisce `True`, ma `IsPosInteger('-1')` restituisce `False`.

5. **(Punti 4)** Si scriva una funzione `ParseDigits(filename)` che legge il file `digits.csv` che trovate nella vostra cartella d'esame. Le informazioni lette dal file devono essere memorizzate in una lista di coppie `Ls`, in cui il primo elemento della coppia è valore della cifra e il secondo valore della coppia è il vettore corrispondente ad un'immagine digitalizzata della cifra corrispondente. La funzione restituisce la lista `Ls`. Per ogni singolo elemento letto file, si usi il predicato `IsPosInteger(w)` per controllare che possa essere convertito in un numero intero positivo. Se un elemento non può essere convertito in un numero intero positivo, il programma deve sollevare un'eccezione di tipo `TypeError`.

6. **(Punti 3)** Si scriva una funzione `Norm(x)` che prende in input un vettore di n elementi $x_1 \ldots, x_n$ (ovvero una lista di n numeri), e ne calcola la norma euclidea:

$$\text{norm}(x) = \sqrt{\sum_{i=1,\ldots,n} x_i^2}$$

   Verrà valutato positivamente l'utilizzo della funzione high-order *reduce*.

7. **(Punti 3)** Scrivere una funzione `Distance(x, y)` che prende in input due vettori $x, y \in \mathbb{R}^n$, e calcola la distanza euclidea tra i due vettori:

$$d(x, y) = \sqrt{\sum_{i=1,\ldots,n} (x_i - y_i)^2}$$

   Verrà valutato positivamente l'utilizzo della funzione high-order *reduce*.

8. **(Punti 4)** Scrivere una funzione `NearestDigit(digit, Ls)` che prende in input un vettore `digit` di un'immagine digitalizzata e una lista di coppie `(cifra, immagine)` come letta dalla funzione `ParseDigits(filename)`. La funzione deve trovare, tra tutte le immagini contenute in `Ls`, quella più simile a `digit`, dove la similitudine è data dalla distanza Euclidea definita al punto precedente (minore è la distanza, maggiore è la similitudine). Trovata l'immagine più simile, la funzione restituisce una tripla (una tupla di 3 elementi), contenente il valore di distanza minimo, il valore della cifra trovata, e il vettore dell'immagine corrispondente (si veda il corpo del `__main__` per un esempio).

9. **(Punti 6)** Scrivere una funzione `Baricentro(As, value)` che data una lista di coppie `(cifra, immagine)`, e il valore di una cifra `value`, esegue le operazioni seguenti.

   (a) Seleziona dalla lista di coppie `As` solo gli elementi corrispondenti a immagini della cifra `value`. Sia `X` la lista cosi ottenuta e sia N la lunghezza di `X`.

   (b) Si implementi una funzione che calcoli il baricentro delle immagini contenute nella lista `X`. Il baricentro rappresenta una "media" di un'insieme di immagini date, ed è definito da una matrice B, che viene calcolata nel modo seguente:

$$b_{ij} = \frac{1}{N} \sum_{k=1,\ldots,N} X(k, i, j)$$

   in cui $X(k, i, j)$ rappresenta il pixel (i, j) dell'immagine k-esima.

   (c) Usare la funzione `DrawMatrix(A)` data nel template di soluzione per visualizzare l'immagine corrispondente al baricentro trovato.

Programmazione 1 **Esame del 3 settembre 2020**

Cognome: **Nome:** **Matricola:** **Postazione:**

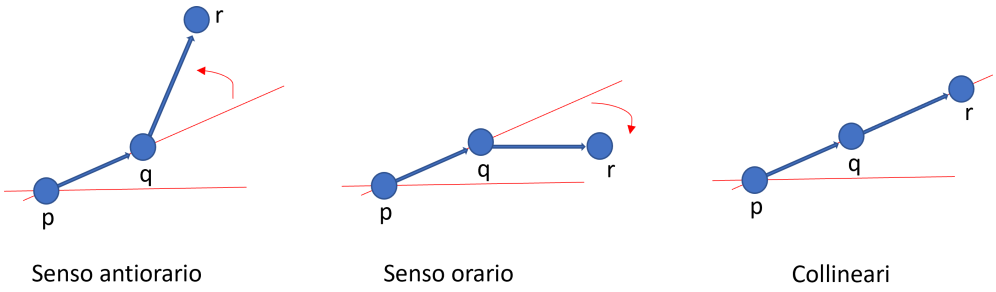
Prima Parte

Nella prima parte dell'esame dovete usare esclusivamente le funzioni ricorsive e le `pairlists`, come studiate nella prima parte del corso. Dovete importare la libreria `pairlists` che viene fornita nella vostra cartella d'esame. Dovete svolgere i seguenti esercizi, completando il file `soluzione_template.py` e **utilizzando le funzioni di test che trovate nel blocco `main` in coda al file**.

1. **(Punti 3)** Scrivere una funzione `Append(As, Bs)` che prende in input le due pairslist `As` e `Bs`, e restituisce una nuova pairslist che contiene, in ordine, prima tutti gli elementi di `As` e poi tutti quelli di `Bs`.

2. **(Punti 4)** Data una tripla ordinata (p, q, r) di 3 punti nel piano, ovvero $p, q, r \in \mathbb{R}^2$, si deve determinare se l'orientamento del vettore $\overrightarrow{qr}$ rispetto al vettore $\overrightarrow{pq}$ sia in verso orario o antiorario, oppure se i due segmenti siano collineari (si veda la figura sotto per un esempio).

   Scrivere una funzione `Orientation(p, q, r)` che prende in input tre coppie di numeri, e restituisce -1 se i due vettori sono in verso antiorario, +1 se sono in verso orario, e 0 se sono collineari. **Suggerimento:** Utilizzare il coefficiente angolare della retta passante per due punti.



Seconda Parte

Nella seconda parte dell'esame potete usare qualsiasi libreria di Python.

Si consideri il file di testo `winereviews.csv`, contenente delle valutazioni di vini. Il file contiene, in ogni riga, le informazioni seguenti: il paese del vino (`country`), il punteggio assegnato da un utente (`points`), il prezzo del vino a bottiglia (`price`), e il tipo di vino (`variety`). Le prima quattro righe del file sono le seguenti:

```
country,points,price,variety
Italy,87,,White Blend
Spain,87,15,Tempranillo-Merlot
Italy,87,16,Frappato
```

Prima di iniziare a svolgere gli esercizi seguenti, si consiglia di leggere attentamente le funzioni di test presenti nel blocco di codice nel `__main__` nel template di soluzioni che vi è stato fornito.

3. **(Punti 4)** Si scriva una classe `Review` che abbia un costruttore che prende in input una lista di stringhe lette dal file, e che inizializza i valori per gli attributi: `country, points, price, variety`. Il primo e ultimo sono degli attributi. Il punteggio è un numero intero da 0 a 100. Il prezzo potrebbe non essere definito, e in questo caso si deve memorizzare la stringa `'nan'`. La classe deve implementare anche i metodi `__str__` e il metodo `__lt__`. Per quest'ultimo operatore di confronto *less-than*, si utilizzi il punteggio della revisione.

4. **(Punti 3)** Si scriva una funzione `ParseReviews(filename)` che legge il file `winereview.csv` che trovate nella vostra cartella d'esame. Le informazioni lette dal file devono essere memorizzate in una lista di oggetti di tipo `Review`. La funzione restituisce la lista `Ls`.

5. **(Punti 2)** Si scriva una funzione `IsFloat(x)` che prende in input un numero `x` e restituisce `True` se il numero può essere convertito in un `float` e se non è uguale alla stringa `'nan'`. Altrimenti la funzione deve restituire `False`. Utilizzare le eccezioni per implementare questa funzione.

6. **(Punti 3)** Scrivere una funzione high-order `Best(Ps, F)` che prende in input una lista di revisioni `Ps` per cui è definito sia il punteggio della revisione che il prezzo, e una funzione `F(x)` che converte una revisione in un valore numerico. La funzione `Best` deve restituire l'elemento di `Ps` che massimizza il valore di `F` corrispondente. La funzione può essere usata nel modo seguente:

   ```
   print("Best:", Best(Ps, lambda x:  x.points/x.price))
   ```

7. **(Punti 3)** Si scriva una funzione high-order `Median(Ps, F)` che prende in input una lista di revisioni `Ps` per cui è definito sia il punteggio della revisione che il prezzo, e una funzione `F(x)` che converte una revisione in un valore numerico. La funzione `Median` deve restituire la revisione di `Ps` che realizza il valore mediano di `F` corrispondente.

8. **(Punti 3)** Si scriva una funzione high-order `Mean(Ps, F)` che prende in input una lista di revisioni `Ps` per cui è definito sia il punteggio della revisione che il prezzo, e una funzione `F(x)` che converte una revisione in un valore numerico. La funzione `Mean` deve restituire il valore medio di `F` per le revisioni in `Ps`.

9. **(Punti 4)** Scrivere una funzione `CountryVariety(Rs)` che prende in input una lista di revisioni, e calcola un dizionario con una chiave per ogni paese (attributo `country`) di cui si ha almeno una revisione. Ad ogni paese viene associato un secondo dizionario con una chiave per ogni tipo di vino `variety`, a cui viene associata una lista con tutti i punteggi associati a quella tipologia di vino per quel paese. In pratica, viene creato e restituito un dizionario di dizionari.

10. **(Punti 3)** Scrivere una funzione `PointVariety(D, n=10)` che prende in input un dizionario come restituito dalla funzione precedente e un numero intero n, e che calcola un nuovo di dizionario di dizionari, con le stesse chiavi (prima il paese e poi la tipologia), ma che associa a ciascuna tipologia che ha ottenuto almeno n valutazioni una coppia di numeri: il punteggio medio e il numero di valutazioni.

Programmazione 1 **Esame del 17 settembre 2020**

Cognome: **Nome:** **Matricola:** **Postazione:**

Prima Parte

Nella prima parte dell'esame dovete usare esclusivamente le funzioni ricorsive e le `pairlists`, come studiate nella prima parte del corso. Dovete importare la libreria `pairlists` che viene fornita nella vostra cartella d'esame. Dovete svolgere i seguenti esercizi, completando il file `soluzione_template.py` e **utilizzando le funzioni di test che trovate nel blocco `main` in coda al file**.

1. **(Punti 2)** Scrivere un predicato `Divisibile(x, n)` che prende in input due numeri interi x e n e restituisce `True` se x è divisibile per n, and `False` altrimenti.

2. **(Punti 4)** Scrivere una funzione `Zip(As, Bs)` che prende in input due pairlists della stessa lunghezza, e restituisce una lista di coppie di elementi, in cui l'ennesimo elemento di `As` é accoppiato con l'ennesimo elemento di `Bs`. Quando le due liste in input non hanno la stessa lunghezza, la funzione deve sollevare una eccezione di tipo `Runtime`.

3. **(Punti 5)** Scrivere una funzione che implementa l'algoritmo di *merge-sort* utilizzando le `pairslist`. Implementate una funzione `MergeSort(Ls, Compare)` che prende in input una lista di numeri interi (lista di tipo `pairslist`), e una predicato di confronto `Compare(x, y)` tra due numeri interi, e restituisce una nuova lista ordinata in base alla funzione di confronto data in input.

   Ad esempio, data una lista di numeri casuali, la funzione può essere usata nel modo seguente:

   ```
   Ls = MakeRandomInts(10, 0, 100)
   print(Ls)
   print(MergeSort(Ls, lambda x,y: x > y))
   ```

Seconda Parte

Nella seconda parte dell'esame potete usare qualsiasi libreria di Python.

Si consideri il file di testo `data.csv`, contenente dei risultati di un questionario sulla mobilità. Il file contiene, in ogni riga, le informazioni seguenti: il corso di laurea dello studente che ha risposto (`Corso`), la risposta alla domanda su quanti giorni passava nella città dove si trova la sua università (`Permanenza`), il mezzo di trasposto usato per raggiungere l'Università (`Trasporto`), il tempo dichiarato per raggiungere l'Ateneo (`Minuti`), e la distanza percorsa (`KM`). Le prima cinque righe del file sono le seguenti:

```
Corso;Permanenza;Trasporto;Minuti;KM
Triennale;Meno di un giorno alla settimana;Treno/Passante;60;76
Triennale;Meno di un giorno alla settimana;Piedi (5 min e oltre);8;1
Triennale;3  giorni;Treno/Passante;15;30
Magistrale;5 o più giorni;Bus;20;3
```

Prima di iniziare a svolgere gli esercizi seguenti, si consiglia di leggere attentamente le funzioni di test presenti nel blocco di codice nel `__main__` nel template di soluzione che vi è stato fornito.

4. **(Punti 1)** Si scriva una funzione `str2int(x)` che prende in input una stringa `x` e, se possibile, converte la stringa x in un numero intero e lo restituisce in output. Se non si può convertire la stringa x in un numero intero, allora la funzione restituisce la stringa `'nan'`.

5. **(Punti 3)** Si scriva una funzione `ParseDati(filename)` che legge il file `data.csv` che trovate nella vostra cartella d'esame. Le informazioni lette dal file devono essere memorizzate in una lista di tuple di 5 elementi (uno per ogni *colonna* del file .csv). Gli ultimi due elementi della tupla (corrispondenti ai Minuti e ai KM di percorrenza) devono essere convertiti in numeri interi utilizzando la funzione implementata nella domanda precedente. La funzione restituisce la lista di tuple.

6. **(Punti 3)** Si scriva una funzione `ComputeMezzi(Ls)` che prende in input una lista di tuple `Ls` letta dal file tramite la funzione dell'esercizio precedente, e restituisce una lista di stringhe, una stringa per ogni tipo di mezzo usato per raggiungere l'Università.

7. **(Punti 4)** Si scriva una funzione high-order `ComputeMean(Ls, F, P)` che prende in input una lista di tuple `Ls`, una funzione `F` che mappa ogni tupla in un valore numerico, e un predicato `P` che prende in input una tupla e restituisce `True` o `False`. La funzione `ComputeMean` deve restituire la media dei valori ottenuti applicando la funzione `F` a ciascuna tupla, ma considerando nel calcolo della media solo quelle tuple per cui il predicato `P` ha restituito `True`.

   Si veda il blocco del `main` per un esempio di utilizzo di questa funzione.

8. **(Punti 4)** Si scriva una funzione `VelocitaMedia(Ls)` che prende in input una lista di tuple `Ls` e restituisce un dizionario. Il dizionario deve contenere una chiave per ogni tipo di mezzo usato per raggiungere l'ateneo (si consiglia di usare la funzione `ComputeMezzi`), e come valore corrispondente la velocità media (in km/h) calcolata utilizzando i tempi di percorrenza e la distanza in km dichiarati nelle risposte.

9. **(Punti 3)** Si scriva una funzione `Permanenza(Ls, n=3)` che prende in input una lista di tuple `Ls` e un numero intero n. La funzione deve restituire una lista ordinata di coppie, dove ogni coppia contiene come primo valore un possibile valore di permanenza dichiarato nelle risposte (ad esempio, si veda la seconda colonna del file di input), e come secondo valore, il numero di studenti che ha dato quella risposta. La lista restituita deve contenere al più n coppie, ordinate in ordine decrescente in base al secondo elemento di ciascuna coppia (ovvero, il numero di studenti che ha dato quella risposta).

10. **(Punti 3)** Si scriva una funzione `CorsoPerm(Ls)` che prende in input una lista di tuple `Ls` e restituisce un dizionario, con una chiave per ogni tipo di corso di studio (dato dalla prima colonna del file di input), e come valore corrispondente la lista ottenuta applicando la funzione `Permanenza(Ls, n=3)` alla sottolista di `Ls` contenente solo le tuple corrispondenti alla chiave considerata.