Si risolva l'esercizio proposto. Se è possibile, includere tutto in un unico file sorgente. Si consiglia di concentrarsi sulla pulizia del codice: pochi punti risolti in modo ordinato saranno valutati meglio di tanti punti trattati disordinatamente. La sufficienza è raggiunta risolvendo correttamente i primi due punti.

Esercizio

Si vuole scrivere una classe `Parabola` che descriva polinomi di grado due a coefficienti reali.

1. Si pongano come membri privati tre variabili corrispondenti ai coefficienti. Tra i membri pubblici, si scriva un opportuno costruttore che richieda come parametri i coefficienti (in modo che il primo sia quello di x^2, il secondo quello di x e il terzo il termine noto).

2. Si implementino il costruttore di copie e l'operatore di assegnazione.

3. Tra i membri pubblici, si scriva l'operatore "+", che restituisce il polinomio somma di due polinomi, e l'operatore "$*$", che restituisce il polinomio prodotto tra un polinomio e un numero reale. Quest'ultimo funziona solo quando il prodotto è scritto in un dato ordine: se `p` è una `Parabola` e `d` è un `double`, funzionerà la sintassi `p*d`. Implementare una funzione ausiliaria (che richiami quella già scritta) in modo che funzioni anche la sintassi `d*p`.

4. Si vuole ora implementare la soluzione delle equazioni di secondo grado. Si scriva una classe `Result` che abbia tra i membri privati una variabile booleana `has_solution` che specifichi se l'equazione ha almeno una soluzione nel campo complesso, una variabile booleana `is_indeterminate` che specifichi se l'equazione ha infinite soluzioni, una variabile reale contenente il valore. Si scriva un opportuno costruttore che richieda tre parametri, con *valori di default* assegnati in modo che `has_solution` sia vero e `is_indeterminate` sia falso se non sono specificati.

5. Si implementi, all'interno della classe `Result`, una funzione void `print` che chieda in ingresso una referenza ad un oggetto di tipo `ostream` (nella libreria `iostream`), e stampi `'Nessuna soluzione'`, `'Infinite soluzioni'` oppure il valore, a seconda dei flag. Si implementi poi

l'overloading dell'operatore `<<` per gli `ostream`, che dovrà servirsi della funzione appena scritta.

6. Si scriva, tra i membri pubblici della classe `Parabola`, una funzione `largest_root` che restituisca un `Result` contenente la parte reale della radice con parte reale più grande. (Tale funzione dovrà essere definita `virtual` per risolvere l'ultimo punto.) Si faccia in modo che i flag vengano impostati opportunamente nel caso in cui l'equazione abbia infinite soluzioni (quando tutti i coefficienti sono nulli) oppure non ne abbia nessuna (quando solo il termine noto è non nullo).

7. Si verifichi il comportamento del codice con il seguente `main`:

```
#include <iostream>
using namespace std;

int main() {
        cout << Parabola(1,1,1).largest_root() << endl;
        cout << Parabola(0,1,1).largest_root() << endl;
        cout << Parabola(0,0,1).largest_root() << endl;
        cout << Parabola(0,0,0).largest_root() << endl;
}
```

8. Si scriva una nuova classe `Parabola_R`, che erediti pubblicamente da `Parabola`, e che, attraverso l'*overriding* della funzione `largest_root`, implementi la seguente funzionalità: il risultato dovrà avere il suo flag `has_solution` falso quando l'equazione non ha soluzioni reali.

9. Per verificare il comportamento polimorfico, si scriva un `main` in cui si istanzia un `vector` di puntatori a oggetti di tipo `Parabola` e lo si riempie con (puntatori a) una `Parabola` e una `Parabola_R`, entrambe inizializzate con il polinomio $x^2 + 1$. Si stampi a schermo per ogni elemento del vettore il risultato della funzione `largest_root`.