Τμήμα ΗΜΜΥ **2ο Εξάμηνο**
Α.Π.Θ. **2019**

ΕΡΓΑΣΙΑ ΣΤΟΝ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

ΠΕΜΠΤΟ ΠΑΡΑΔΟΤΕΟ (ΠΡΟΑΙΡΕΤΙΚΟ)

Υλοποίηση έξυπνου παίκτη για το παιχνίδι σε C++

Το πέμπτο παραδοτέο της εργασίας αφορά τη δημιουργία ενός έξυπνου παίκτη για το παιχνίδι της Φοιτητούπολης. Σημειώστε ότι **το παραδοτέο είναι προαιρετικό και δε προσμετράται στις μονάδες της εργασίας.** Αντί αυτού, οι υλοποιήσεις σας θα συμμετάσχουν σε ένα τουρνουά! Οι παίκτες της νικήτριας ομάδας θα πάρουν **bonus 1.5 μονάδας**, οι παίκτες της δεύτερης καλύτερης ομάδας θα πάρουν **bonus 1.0 μονάδας** και οι παίκτες της τρίτης καλύτερης ομάδας (που θα προκύψει από μικρό τελικό) θα πάρουν **bonus 0.5 μονάδας**.

Σημειώστε ότι **για να φτιάξετε έναν έξυπνο παίκτη θα πρέπει να έχετε υλοποιήσει πρώτα το τέταρτο παραδοτέο της εργασίας**. Πριν λοιπόν ξεκινήσετε, θα πρέπει να **αντικαταστήσετε τα αρχεία card.cpp και shuffle.h** με τα υλοποιημένα αρχεία σας από το τέταρτο παραδοτέο.

Στη συνέχεια, θα πρέπει να υλοποιήσετε τον παίκτη σας στην κλάση SmartPlayer στο αρχείο **smartplayer.cpp**. Στο συγκεκριμένο αρχείο θα πρέπει αρχικά να θέσετε το όνομα του παίκτη στον constructor ανάλογα με τον αριθμό της ομάδας σας χρησιμοποιώντας τρία ψηφία (π.χ. για την ομάδα 4 θα πρέπει να βάλετε όνομα "Team 004"). Στη συνέχεια, θα πρέπει να υλοποιήσετε τις εξής συναρτήσεις αποφάσεων για τον έξυπνο παίκτη:

1. Συνάρτηση decideBuy
   Η συνάρτηση αυτή λαμβάνει ως είσοδο ένα συγκεκριμένο τετράγωνο του παιχνιδιού και επιστρέφει **true** αν ο παίκτης σας επιθυμεί (με βάση τη στρατηγική σας) **να αγοράσει το συγκεκριμένο τετράγωνο** ή **false** αν **δεν το επιθυμεί**.
2. Συνάρτηση decideUpgrade
   Η συνάρτηση αυτή επιστρέφει έναν ακέραιο αριθμό που είναι το **id του τετραγώνου** που ο έξυπνος παίκτης σας **επιθυμεί να αναβαθμίσει**. Αν ο παίκτης **δεν επιθυμεί να αναβαθμίσει** κάποιο τετράγωνο, τότε η συνάρτηση **επιστρέφει -1**.

Υπόδειξη: για να κατασκευάσετε τη στρατηγική σας, προτείνεται να χρησιμοποιήσετε τα spaces και opponent που είναι προσβάσιμα μέσω κληρονομικότητας από την κλάση Player. Το spaces είναι ένας πίνακας από δείκτες σε τετράγωνα (αντικείμενα Space) του παιχνιδιού, ενώ το opponent είναι ο δείκτης στον αντίπαλο παίκτη (αντικείμενο Player). Ο παίκτης **απαγορεύεται να πειράξει αυτά τα αντικείμενα (π.χ. απαγορεύεται η χρήση συναρτήσεων set)**, μπορεί ωστόσο να δει την κατάστασή τους και ανάλογα να πάρει τις κατάλληλες αποφάσεις (π.χ. βλέπω ότι ο αντίπαλός μου έχει σχεδόν αγοράσει μια περιοχή, οπότε αγοράζω ένα από τα τετράγωνα ώστε να μην του επιτρέψω να χτίσει).

Σημειώνετε ότι οι παραπάνω συναρτήσεις **δεν υλοποιούν κάποια αγορά ή αναβάθμιση (αυτό απαγορεύεται καθώς αποτελεί επέμβαση στα αντικείμενα)**. Επίσης, **δεν ελέγχουν αν μια αγορά ή αναβάθμιση είναι έγκυρη** (μπορείτε φυσικά να το ελέγξετε αλλά δεν είναι αυτός ο σκοπός τους). Επιστρέφουν μόνο αυτά που αναφέρονται παραπάνω (η decideBuy ένα true/false και η decideUpgrade έναν ακέραιο). Οι υλοποιήσεις και οι έλεγχοι των αγορών/αναβαθμίσεων υλοποιούνται σε άλλες συναρτήσεις.

Υπόδειξη: για δική σας ευκολία μπορείτε μέσα στις παραπάνω συναρτήσεις να χρησιμοποιήσετε τις συναρτήσεις canBuy και canUpgrade της κλάσης Player, και γενικά όποιες συναρτήσεις επιθυμείτε **αρκεί να μην επηρεάζετε τα αντικείμενα**.

Για να δοκιμάσετε τον παίκτη σας, **θα πρέπει να αλλάξετε κατάλληλα τις εντολές στις γραμμές 52 και 53 του αρχείου main.cpp**. Στις συγκεκριμένες γραμμές μπορείτε π.χ. να δοκιμάσετε τον παίκτη σας έναντι του απλού ComputerPlayer ως εξής:

```
players[0] = new ComputerPlayer(0);
players[1] = new SmartPlayer(1);
```

Επίσης, μπορείτε π.χ. να δοκιμάσετε να παίξετε εσείς οι ίδιοι αντίπαλοι με τον παίκτη σας ως εξής:

```
players[0] = new HumanPlayer(0);
players[1] = new SmartPlayer(1);
```

Μπορείτε φυσικά να κάνετε αλλαγές και στον ComputerPlayer, ώστε να δείτε πώς ο παίκτης αντιμετωπίζει άλλες στρατηγικές (ή ακόμα και να αρχικοποιήσετε και τους δύο παίκτες με new SmartPlayer για να δείτε πώς ο παίκτης τα καταφέρνει ενάντια στον εαυτό του).

Παρατηρήσεις

Η υλοποίηση θα πρέπει να γίνει στη C++ και να μπορεί να ανοίξει με το CodeBlocks, **με τις εκδόσεις που χρησιμοποιούμε** στο πλαίσιο του μαθήματος. Ο κώδικάς σας θα πρέπει να είναι καλά τεκμηριωμένος, ώστε να είναι παντού σαφείς οι λεπτομέρειες υλοποίησης.
Για την υλοποίηση, σας δίνονται τα αρχεία κεφαλίδων .h των κλάσεων/συναρτήσεων που πρέπει να υλοποιήσετε καθώς και κάποιες βοηθητικές κλάσεις/συναρτήσεις. Επιπλέον, σας δίνεται ο κώδικας της συνάρτησης main (main.cpp). Σε καμία περίπτωση δεν επιτρέπεται να επέμβετε στον κώδικα των κλάσεων και των συναρτήσεων αυτών. Σε περίπτωση που το κάνετε, η εργασία σας αυτομάτως θεωρείται λανθασμένη και μηδενίζεται. Θα πρέπει μόνο να αντιγράψετε τα αρχεία card.cpp και shuffler.h από το τέταρτο παραδοτέο σας και να γράψετε κώδικα μόνο στο αρχείο smartplayer.cpp.

Παραδοτέο

Το παραδοτέο θα είναι **ένα αρχείο zip με όνομα Monopoly.zip** που θα περιλαμβάνει **όλο το project (τα αρχεία που θα υλοποιήσετε αλλά και αυτά που σας έχουν δοθεί)**, δηλαδή **ακριβώς ίδιο με το αρχείο Monopoly.zip που δίνεται**, φυσικά **με τον κώδικα υλοποιημένο**.

Προθεσμία υποβολής

Το παραδοτέο πρέπει να παραδοθεί μέχρι τις **23:59 της Τρίτης 28 Μαΐου**. Καμία παρέκκλιση δε θα γίνει από την παραπάνω προθεσμία.