Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
https://helios.ntua.gr/course/view.php?id=869

Προγραμματιστικές Τεχνικές

Οι ασκήσεις αυτής της σειράς περιστρέφονται γύρω από μία γενίκευση του παιχνιδιού Π_n του Specker, που γνωρίσατε στο πρώτο εξάμηνο (αν διαβάζατε τις πίσω σελίδες των εκφωνήσεων των ασκήσεων).

Σε αυτή τη σειρά ασκήσεων θα υλοποίησετε αντικείμενα που συμμετέχουν στο παιχνίδι και αλληλεπιδρούν μεταξύ τους. Τα αντικείμενα αυτά είναι οι *κινήσεις*, η *κατάσταση* του παιχνιδιού, οι *παίκτες* και η ίδια η *εκτέλεση του παιχνιδιού*.

Γενίκευση παιχνιδιού Specker: Περιγραφή

Στη γενίκευση που μας ενδιαφέρει, στο παιχνίδι συμμετέχουν p παίκτες (αριθμημένοι από 0 έως $p-1$) και υπάρχουν n σωροί με ομοειδή κέρματα (αριθμημένοι από 0 έως $n-1$). Ο παίκτης 0 παίζει πρώτος, μετά ο παίκτης 1, κ.ο.κ., εως τον παίκτη $p-1$, μετά τον οποίο παίζει πάλι ο παίκτης 0, κυκλικά.

Σε κάθε κίνηση, ο παίκτης που παίζει επιλέγει δύο σωρούς, έστω το σωρό i και το σωρό j, αφαιρεί $k > 0$ κέρματα από το σωρό i (αυτό φυσικά πρέπει να είναι δυνατό να συμβεί, δηλαδή ο σωρός i πρέπει να έχει τουλάχιστον k κέρματα) και βάζει m κέρματα (όπου $0 \leq m < k$) στο σωρό j. Προσέξτε ότι αν επιλέξει $m = 0$ (δηλαδή να μη βάλει κανένα κέρμα), τότε η επιλογή του σωρού j δεν είναι σημαντική.

Το παιχνίδι τελειώνει όταν αφαιρεθούν όλα τα κέρματα από όλους τους σωρούς και ο παίκτης που κερδίζει είναι ο τελευταίος που έπαιξε.

Άσκηση 4 Γενίκευση παιχνιδιού Specker: Κινήσεις και κατάσταση

Προθεσμία υποβολής στον grader: 23/3/2025

Υλοποιήστε την κλάση `Move` για την αναπαράσταση των *κινήσεων* του παιχνιδιού. Η κλάση αυτή πρέπει να υποστηρίζει τις εξής λειτουργίες, τις οποίες πρέπει να υλοποιήσετε:

```
class Move {
public:
  // Take sc coins from heap sh and put tc coins to heap th.
  Move(int sh, int sc, int th, int tc);

  int getSource() const;
  int getSourceCoins() const;
  int getTarget() const;
  int getTargetCoins() const;

  friend ostream & operator << (ostream &out, const Move &move);
};
```

Η εκτύπωση των κινήσεων πρέπει να εμφανίζει τα εξής (χωρίς αλλαγή γραμμής), ανάλογα με το αν οι κινήσεις βάζουν ή όχι πίσω νομίσματα:

```
takes 9 coins from heap 2 and puts 8 coins to heap 0
takes 3 coins from heap 1 and puts nothing
```

Στη συνέχεια, υλοποιήστε την κλάση `State` για την αναπαράσταση των *καταστάσεων* του παιχνιδιού. Η κλάση αυτή πρέπει να υποστηρίζει τις εξής λειτουργίες:

```
class State {
public:
  // State with h heaps, where the i-th heap starts with c[i] coins.
  // A total of n players are in the game, numbered from 0 to n-1,
  // and player 0 is the first to play.
  State(int h, const int c[], int n);
  ~State();

  void next(const Move &move);  // may throw logic_error
  bool winning() const;

  int getHeaps() const;
  int getCoins(int h) const;    // may throw logic_error

  int getPlayers() const;
  int getPlaying() const;

  friend ostream & operator << (ostream &out, const State &state);
};
```

Η μέθοδος `next` εκτελεί την κίνηση `move` και έχει ως αποτέλεσμα την (επί τόπου) αλλαγή της τρέχουσας κατάστασης. Αν η κίνηση είναι άκυρη (π.χ., κάποιος σωρός είναι εκτός ορίων ή δεν υπάρχουν τα απαιτούμενα κέρματα), τότε πρέπει να εγείρεται μία εξαίρεση της κλάσης `logic_error` που ορίζεται στη standard library της C++ και βρίσκεται στο αρχείο επικεφαλίδας `<stdexcept>`, π.χ.

```
if (h < 0 || h >= maxHeaps) throw logic_error("invalid heap");
```

Η εκτύπωση μίας κατάστασης πρέπει να εμφανίζει τα πλήθη των κερμάτων κάθε σωρού, διαδοχικά, χωρισμένα με κόμματα, ακολουθούμενα από τον παίκτη που έχει σειρά να παίξει και το συνολικό πλήθος των παικτών, ως εξής (χωρίς αλλαγή γραμμής):

```
0, 20, 17 with 0/5 playing next
```

Στο αρχείο που θα ανεβάσετε στον grader θα πρέπει να συμπεριλάβετε (μόνο) τις δηλώσεις των κλάσεων `Move` και `State` και τις υλοποιήσεις των μεθόδων τους.

Άσκηση 5 Γενίκευση παιχνιδιού Specker: Παίκτες

Προθεσμία υποβολής στον grader: 23/3/2025

Υλοποιήστε την αφηρημένη κλάση `Player` για την αναπαράσταση των *παικτών* του παιχνιδιού. Η κλάση αυτή πρέπει να υποστηρίζει τις εξής λειτουργίες, τις οποίες πρέπει να υλοποιήσετε:

```
class Player {
public:
  Player(const string &n);
  virtual ~Player();

```

```
  virtual const string & getType() const = 0;
  virtual Move play(const State &s) = 0;

  friend ostream & operator << (ostream &out, const Player &player);
};
```

Κάθε παίκτης (δηλαδή κάθε αντικείμενο μίας κλάσης που παράγεται από την `Player`) χαρακτηρίζεται από το *όνομα* και τον *τύπο* του. Το όνομα πρέπει να αποθηκεύεται στην κλάση `Player`. Ο τύπος είναι ό,τι επιστρέφει η αφηρημένη μέθοδος `getType` που θα ορίζεται στις υποκλάσεις της `Player`. Η εκτύπωση θα πρέπει να εκτυπώνει τον τύπο (π.χ., "Greedy") και το όνομα (π.χ., "Alan") ενός παίκτη ως εξής (χωρίς αλλαγή γραμμής):

```
Greedy player Alan
```

Η αφηρημένη μέθοδος `play` πρέπει να επιστρέφει την κίνηση που θα παίξει κάποιος παίκτης σε κάποια κατάσταση του παιχνιδιού. Υλοποιήστε τέσσερις τύπους παικτών, που να ορίζουν τη μέθοδο αυτή ώστε να λειτουργεί (ντετερμινιστικά!) με τον εξής τρόπο:

1. Τύπος "Greedy": επιλέγει το σωρό με τα περισσότερα κέρματα, τα αφαιρεί όλα και δεν βάζει κανένα πίσω. Αν υπάρχουν περισσότεροι σωροί με τον ίδιο μέγιστο αριθμό κερμάτων, επιλέγεται αυτός με τον μικρότερο αύξοντα αριθμό. Υλοποιήστε την κλάση `GreedyPlayer`.

2. Τύπος "Spartan": επιλέγει το σωρό με τα περισσότερα κέρματα, αφαιρεί μονο ένα και (προφανώς) δεν βάζει κανένα πίσω. Αν υπάρχουν περισσότεροι σωροί με τον ίδιο μέγιστο αριθμό κερμάτων, επιλέγεται αυτός με τον μικρότερο αύξοντα αριθμό. Υλοποιήστε την κλάση `SpartanPlayer`.

3. Τύπος "Sneaky": επιλέγει το σωρό με τα λιγότερα κέρματα, τα αφαιρεί όλα και δεν βάζει κανένα πίσω. Αν υπάρχουν περισσότεροι σωροί με τον ίδιο ελάχιστο αριθμό κερμάτων, επιλέγεται αυτός με τον μικρότερο αύξοντα αριθμό. Υλοποιήστε την κλάση `SneakyPlayer`.

4. Τύπος "Righteous": επιλέγει το σωρό με τα περισσότερα κέρματα. Αν αυτός έχει c κέρματα, αφαιρεί $\lceil \frac{c}{2} \rceil$ και βάζει $\lceil \frac{c}{2} \rceil - 1$ στο σωρό με τα λιγότερα κέρματα. Αν υπάρχουν περισσότεροι σωροί με τον ίδιο μέγιστο ή ελάχιστο αριθμό κερμάτων, επιλέγονται αντίστοιχα αυτοί με τον μικρότερο αύξοντα αριθμό. Υλοποιήστε την κλάση `RighteousPlayer`.

Στο αρχείο που θα ανεβάσετε στον grader θα πρέπει να συμπεριλάβετε (μόνο) τις δηλώσεις της κλάσης `Player` και των παραγόμενων κλάσεων αυτής, καθώς και τις υλοποιήσεις των μεθόδων τους.

Άσκηση 6 Γενίκευση παιχνιδιού Specker: Το παιχνίδι

Προθεσμία υποβολής στον grader: 23/3/2025

Υλοποιήστε την αφηρημένη κλάση `Game` για την αναπαράσταση του ίδιου του παιχνιδιού. Η κλάση αυτή πρέπει να υποστηρίζει τις εξής λειτουργίες, τις οποίες πρέπει να υλοποιήσετε:

```
class Game {
public:
  Game(int heaps, int players);
  ~Game();

  void addHeap(int coins);              // may throw logic_error
  void addPlayer(Player *player);       // may throw logic_error
  void play(ostream &out);              // may throw logic_error
```

```
9
10    int getPlayers() const;
11    const Player *getPlayer(int p) const;  // may throw logic_error
12  };
```

Οι μέθοδοι `addHeap` και `addPlayer` προσθέτουν έναν-έναν τους σωρούς του παιχνιδιού (δηλαδή τον αρχικό αριθμό κερμάτων) και τους παίκτες, με τη σειρά. Η μέθοδος `play` πρέπει να παίζει το παιχνίδι μέχρι κάποιος παίκτης να κερδίσει και να εκτυπώνει στη ροή εξόδου `out` την εξέλιξη του παιχνιδιού. Μια δεύτερη κλήση της μεθόδου `play` θα πρέπει να παίζει το ίδιο παιχνίδι, από την αρχή. Αν οι παίκτες συμπεριφέρονται ντετερμινιστικά, το αποτέλεσμα θα πρέπει να είναι ακριβώς το ίδιο. Επίσης, αν εγείρεται εξαίρεση `logic_error`, το παιχνίδι πρέπει να σταματάει και να εκτυπώνεται το μήνυμα της εξαίρεσης.

Για παράδειγμα, το πρόγραμμα:

```
1  int main() {
2    Game specker(3, 4);
3    specker.addHeap(10);
4    specker.addHeap(20);
5    specker.addHeap(17);
6    specker.addPlayer(new SneakyPlayer(”Tom”));
7    specker.addPlayer(new SpartanPlayer(”Mary”));
8    specker.addPlayer(new GreedyPlayer(”Alan”));
9    specker.addPlayer(new RighteousPlayer(”Robin”));
10   specker.play(cout);
11 }
```

θα πρέπει να εκτυπώνει τα παρακάτω:

```
1  State: 10, 20, 17 with 0/4 playing next
2  Sneaky player Tom takes 10 coins from heap 0 and puts nothing
3  State: 0, 20, 17 with 1/4 playing next
4  Spartan player Mary takes 1 coins from heap 1 and puts nothing
5  State: 0, 19, 17 with 2/4 playing next
6  Greedy player Alan takes 19 coins from heap 1 and puts nothing
7  State: 0, 0, 17 with 3/4 playing next
8  Righteous player Robin takes 9 coins from heap 2 and puts 8 coins to heap 0
9  State: 8, 0, 8 with 0/4 playing next
10 Sneaky player Tom takes 8 coins from heap 0 and puts nothing
11 State: 0, 0, 8 with 1/4 playing next
12 Spartan player Mary takes 1 coins from heap 2 and puts nothing
13 State: 0, 0, 7 with 2/4 playing next
14 Greedy player Alan takes 7 coins from heap 2 and puts nothing
15 State: 0, 0, 0 with 3/4 playing next
16 Greedy player Alan wins
```

Στο αρχείο που θα ανεβάσετε στον grader θα πρέπει να συμπεριλάβετε (μόνο) τη δήλωση της κλάσης `Game` και τις υλοποιήσεις των μεθόδων της.

Άσκηση 7 Γενίκευση παιχνιδιού Specker: Και τώρα παίζουμε

Προθεσμία παράδοσης: 29/6/2025 — BONUS

Υλοποιήστε μία κλάση παίκτη με όνομα `pi24bXYZ` (το username σας στο εργαστήριο) και ό,τι όνομα τύπου θέλετε, η οποία να παίζει το παιχνίδι με όποια στρατηγική εσείς επιθυμείτε. Στο τέλος του εξαμήνου, θα γίνει ένα τουρνουά 1000 παιχνιδιών με τυχαίες αρχικές καταστάσεις, στο οποίο θα συμμετέχουν όλοι όσοι έχουν υποβάλει λύσεις.

Στείλτε την υλοποίησή σας με e-mail στο `progtech@courses.softlab.ntua.gr`.