שאלה 1 – תכן בשפת C++

המשחקים האולימפיים (Olympics) שיתקיימו בפריז בשנת 2024 יציגו מגוון רחב של תחומי ספורט.

בשאלה זו נתמקד בספורט השחייה, ונכתוב (באופן חלקי) ממשק שיודע לנהל תחרות שחייה ברמה אולימפית.

לכל שחיין (Swimmer) באולימפיאדה יש רמת אנרגיה (מספר חיובי), אוסף של משחים שהוא יודע לשחות, וטקטיקת שחייה (Tactic) לפיה הוא שוחה את כל המשחים שלו.

שחיין מקצועי נח טוב מאוד ולכן הוא מזנק לכל מקצה שלו עם רמת אנרגיה (energy) ששווה לרמת האנרגיה הכי גבוהה האפשרית עבורו (MaxEnergy), בלי קשר לרמת האנרגיה שהייתה לו בסוף המשחה הקודם שהיה לו.

משחה (Event) מורכב ממרחק במטרים (Distance) שהוא מספר טבעי ומסגנון (Stroke). הסגנונות הם: פרפר (Fly), גב (Back), חזה (Breast), חתירה (Free), ומעורב אישי (IM – Individual Medley).

- בסגנונות פרפר, גב, וחזה ניתן לשחות רק 100 ו-200 מטרים.
- בסגנון החתירה ניתן לשחות 50, 100, 200, 400, 800 ו-1500 מטרים.
- בסגנון מעורב אישי ניתן לשחות רק 200 ו-400 מטרים.

הערות:

- משחה לא יכול להופיע פעמיים באוסף המשחים של שחיין.
- באולימפיאדה מתקיימים גם מקצי שליחים שונים, נתעלם מהם בשאלה זו.

טקטיקה (Tactic) היא האופן שבו שחיין מחלק את הכוח שלו במשחה, ולכל טקטיקה ישנה פונקציית איבוד אנרגיה הנקראת swim. הטקטיקות הקיימות הן:

- ספרינט (Sprint) – השחיין מתחיל לשחות בכל הכוח מההתחלה ומתעייף מאוד אחרי 100 מטרים, דבר שגורם לו לשחות את המשך המקצה שלו (כל מה שאחרי 100 המטרים הראשונים) לאט מאוד. לכן ספרינטר אינו מסוגל לשחות משחים מעל 100 מטר.
  פונקציית איבוד האנרגיה:
  $$Swim_{\text{Sprint}}(Distance) = \sqrt{\frac{Distance}{100}}$$
- התחלה איטית וספרינט בסיום (BackEnd) – השחיין מתחיל לשחות את המשחה לאט, שומר כוח במשך רוב המשחה, ומגביר מאוד את הקצב לקראת הסיום. לכן מותר לו לשחות את כל המרחקים האפשריים. פונקציית איבוד האנרגיה:
  $$Swim_{\text{BackEnd}}(Distance) = \left(1-\left(\frac{Distance-2600}{2600}\right)^2\right)$$
- שמירה על קצב קבוע (Pace) – השחיין שוחה את כל המשחה בקצב אחיד ולכן גם לו מותר לשחות את כל המרחקים האפשריים. פונקציית איבוד האנרגיה:
  $$Swim_{\text{Pace}}(Distance) = \left(\frac{Distance}{1500}\right)$$

נגדיר את איבוד האנרגיה של שחיין עם טקטיקה Tactic להיות:

$$energy_{\text{loss}} = MaxEnergy \cdot Swim_{\text{Tactic}}(Distance)$$

לצורך הפשטות, נניח ששחיין שאיבד יותר אנרגיה שחה מהר יותר (כי הוא התאמץ יותר). לפיכך נגדיר את השחיין המנצח בתור השחיין שאיבד הכי הרבה אנרגיה.

*אם המשוואות הנ"ל מסובכות מידי עבורכם או שאתם לא מבינים את מקורן ולמה הן שקולות להגדרות המילוליות – תתמודדו.

לפניכם ממשק חלקי של המחלקה Tactic:

```
class Tactic {
public:
    Tactic() = default;
    virtual ~Tactic() = default;
    virtual shared_ptr<Tactic> clone() const = 0;
    virtual float swim(int distance) const = 0; //Swim_Tactic(Distance)
};
```

א. ממשו את שלוש הטקטיקות המפורטות מעלה.
יש לממש כל פונקציית עזר וכל מחלקת עזר שאתם מוסיפים.
אין להוסיף פונקציות לממשק המחלקה, אך ניתן להוסיף מתודות פרטיות.
ניתן להשתמש בפונקציה pow(num, 2) כדי לחשב ריבוע של מספר, ובפונקציה sqrt(num) כדי לחשב שורש של מספר. שתי הפונקציות מחזירות float.

לפניכם ממשק חלקי של המחלקה Event:

```
enum Stroke {Fly, Back, Breast, Free, IM};

class Event {
    static const int DEFAULT_DIST = 50;
    static const Stroke DEFAULT_STROKE = Free;
public:
    Event(int distance =DEFAULT_DIST, Stroke stroke =DEFAULT_STROKE);
    Stroke getStroke() const;
    int getDistance() const;
};
```

ב. ממשו את המחלקה Event.

יש לממש כל פונקציית עזר וכל מחלקת עזר שאתם מוסיפים.
אין להוסיף פונקציות לממשק המחלקה, אך ניתן להוסיף מתודות פרטיות.
יש לבדוק שהמשחה שנוצר הוא רשמי.

- משחה נחשב לא רשמי אם המרחק שלו לא קיים כלל או אם המרחק שלו לא מתאים לסגנון.
- בשני המקרים יש לזרוק חריגת InvalidEvent.

המתודות getStroke, getDistance מחזירות את הסגנון ואת המרחק של המקצה בהתאמה.

ג. שרטטו תכן העונה לצרכי כל המערכת בעזרת UML. ציינו את שמה של כל תבנית תכן שהשתמשתם בה.

לפניכם ממשק חלקי של המחלקה Swimmer:

```
class Swimmer {
public:
    Swimmer(float energy, const Tactic& tactic, set<Event> events);

    bool canSwim(const Event& event) const;
    float swim(const Event& event) const;
    // returns the amount of energy lost from the swim (energy_loss)

    bool addEvent(const Event& event);
};
```

הערות:

- ניתן להניח שהמחלקה Event מקיימת את הדרישות עבור Set.

ד. ממשו את המחלקה Swimmer.

יש לבדוק שהשחיין שנוצר הוא תקין.

- רמת האנרגיה המקסימלית של שחיין חייבת להיות חיובית.
- כל שחיין חייב לדעת לשחות לפחות משחה אחד.
- כפי שכבר נאמר, ספרינטר לא יכול לשחות משחים שאורכם מעל 100 מטרים.
- אם אחד מהתנאים הנ"ל מופר יש לזרוק חריגת InvalidSwimmer.

המתודה canSwim מחזירה האם השחיין יכול לשחות את event.
המתודה swim מחזירה את כמות האנרגיה שירדה.

- אם השחיין לא יכול לשחות את event המתודה תחזיר 0.

המתודה addEvent מוסיפה משחה לאוסף המשחים שהשחיין יודע לשחות, ומחזירה האם היא הצליחה.

- לא ניתן להוסיף משחה שכבר קיים.
- כפי שכבר נאמר, עבור ספרינטרים לא ניתן להוסיף שום משחה שאורכו מעל 100 מטרים.

לפניכם ממשק חלקי ומימוש חלקי של המחלקה Olympics:

```
class Olympics {
public:
    Olympics(const set<Swimmer>& swimmers);
    set<Swimmer> Race(const Event& event) const;
};
```

נתון לכם שהמחלקה Swimmer מקיימת את הדרישות ל-set.
כמו כן, ניתן להניח שאין כפילויות בשחיינים (לא צריך לבדוק זאת).

ה. ממשו את המתודה Race של מחלקת Olympics המקבלת משחה ומחזירה את השחיין/קבוצת השחיינים שזכה/זכו במדליית הזהב. שימו לב שייתכן ולא כל השחיינים יכולים להתחרות במשחה.