תכנות מונחה עצמים
עבודת הגשה 4

מועד הגשה: 16/5/2019בשעה 23:50

הוראות הגשה:

1. **אנא קראו בעיון את כל תיאור העבודה בטרם תתחילו לכתוב קוד.**
2. הגשה באופן עצמאי בלבד. הגשה בקבוצות תוביל לציון 0 בעבודה.
3. אין לשתף או להעתיק את העבודה או חלקים ממנה. עבירה על הוראה זו תוביל לציון 0 בעבודה.
4. הגשה דרך מערכת מודול בלבד. **שום עבודה לא מתקבלת במייל**!
5. יש למקם כל מחלקה שיהיה עליכם ליצור, בשני קבצים נפרדים H ו-CPP. יש להכניס את החלקה התיאורטי בקובץ וורד נפרד. יש להכניס את כל הקבצים של החלקה המעשי + קובץ הוורד לתיקיה אחת, ואז לכווץ יחד. נדרש להגיש קובץ אחד בפורמט RAR או ZIP המכיל את כל הקבצים של כל השאלות. לקובץ המכווץ יהיה שם המהווה את מספר ת.ז. של המגיש.
6. **שאלות ובקשות בקשר לעבודה להפנות אך ורק לאחראית התרגיל, יעל וקסלר, במייל: yaelva@ac.sce.ac.il.**

חלק א' – תאורטי (מענה בקובץ טקסט – וורד): 12 נקודות

1. *2 נק'* מהם השגיאות בקוד :

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

    class A{
    private:
            int x;
    public:
            A():x(0){}
            void SetX(int& x){ this->x = x; }
            void Print()const{ cout << x << endl; }
            virtual void func1() const = 0;
    };

    class B :public A{
    public:
            void func1(){ cout << "B::func1" << endl; }
            virtual void func2() { cout << "B::func2" << endl; };
    };

    class C :public A{
    public:
            void func1() const { cout << "C::func1" << endl; }
            void func2() const { cout << "C::func2" << endl; }
    };

    int main()
    {
            A* a1 = new B();
            a1->func1();
            C* c = new C();
            c->func1();
            c=a1;
            c->SetX(1);
```

```
        return 0;
}
```

2. *2 נק'* באילו מקרים מחלקה יורשת חייבת לממש את כל הפונקציות הווירטואליות של מחלקת הבסיס?
3. *2 נק'* אם ההורסים (dtor) של מחלקת הבסיס אינו וירטואלי אך ובמחלקה הנגזרת ההורס כן וירטואלי אזי בעת מחיקת המחלקת הנגזרת – ההורס של מחלקת הבסיס לא יופעל. האם הטענה נכונה / לא נכונה ? יש לנמק.
4. *6 נק'* נתון הקוד הבא:

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

class A{
   public:
  A() {cout<<"A::C'tor"<<endl;}
  A(const A& a) {cout<<"A::Copy C'tor"<<endl;}
  virtual ~A() {cout<<"A::D'tor"<<endl;}
  virtual void func1() {cout<<"A::func1()"<<endl;}
  virtual void func2() =0;
   void func3() {cout<<"A::func3()"<<endl; func1();}
};

class B1 :virtual public A{
    public:
    B1() {cout<<"B1::C'tor"<<endl;}
    B1(const B1& b1) : A(b1) {cout<<"B1::Copy C'tor"<<endl;}
    ~B1() {cout<<"B1:D'tor"<<endl;}
    void func1() {cout<<"B1::func1()"<<endl;}
    virtual void func2() {cout<<"B1::func2()"<<endl; func3();}
};

class B2: virtual public A {
    public:
      B2() {cout<<"B2::C'tor"<<endl;}
     B2(const B2& b2) : A(b2) {cout<<"B2::Copy C'tor"<<endl;}
     ~B2() {cout<<"B2::D'tor"<<endl;}
     virtual void func3() {func2(); cout<<"B2::func3()"<<endl;}
};

class C :public B1, public B2{
   public:
   C () {cout<<"C::C'tor"<<endl;}
   C (const C& c){cout<<"C::Copy C'tor"<<endl;}
   ~C() {cout<<"C::D'tor"<<endl;}
   virtual void func1() {cout<<"C::func1()"<<endl;}
   void func2() const {cout<<"C::func2()"<<endl;}
};
```

א. מלאו את הטבלאות הוירטואליות של המחלקות הנ"ל.

A class VTABLE	B1 class VTABLE	B2 class VTABLE	C class VTABLE

שימו לב: חובה לרשום לפני כל מתודה לאיזו מחלקה היא שייכת ע"י הוספת הקידומת ::ClassName לפני כל מתודה, כמו כן, **יש לשמור על הסדר הנכון** בטבלאות הוירטואליות.

א. מה הפלט המלא של קטע הקוד הבא:

```
int main()
{
   A* a = new B1();
   a->func2();
   B2* b2 = new C();
   b2->func3();
   delete a;
   delete b2;
   return 0;
}
```

חלק ב' – מעשי (ההגשה היא של קבצי ה- CPP ו-H בלבד) – 100 נקודות:

רקע:

בעבודה זו, עליכם לממש מערכת ניהול ליגת העל לשנת 2019 המשבצת שחקנים וצוותים מקצועיים לקבוצות ומנהלת את מערך המשחקים.

לצורך בניית הפתרון עליכם להשתמש במספר מחלקות שיפורטו בהמשך, אשר ***היחסים*** בניהן יכולים להיות : ***הכלה*** , ***הורשה***. עליכם לבחור את היחסים הרלוונטיים בהתאם לנתונים שקיבלתם.

במימוש המחלקות , יש להשתמש בירושה בצורה יעילה (כלומר , מחלקת הבן נעזרת במתודות של מחלקת האב למימוש המתודה שלה – אם זה אפשרי) ובמקומות הנחוצים יש להשתמש במחלקות אבסטרקטיות במקומות הנכונים.

הקפידו על תכנות נכון reference , const , encapsulation (וכו')

אין להשתמש באובייקטים גלובליים.

הפתרון יש לממש בעזרת פולימורפיזם ו -RTTI.

- ✓ בליגת העל משתתפות 5 קבוצות. בכל קבוצה יש כמה סוגים של שחקנים (לפחות 2 שוערים, 3 חלוצים, 4 קשרים, 3 מגנים, 3 בלמים) , מאמן ומנהל.
- ✓ המשחק מתקיים באיצטדיון מסויים ונשפטים ע"י 3 שופטים.
- ✓ במהלך העונה כל קבוצה משחקת נגד כל אחת מהקבוצות האחרות בליגה, פעם אחת באצטדיונה הביתי ופעם אחת באצטדיון היריבה. (סה"כ 8 משחקים)
- ✓ סה"כ משחקים בליגה 20.
- ✓ המשכורת בסיס לכל שחקן היא 6500=***basis*** .
- ✓ המשכורת בסיס לכל עובד שהוא לא שחקן היא 4500=***basis*** .
- ✓ כל שחקן יכול לשחק בקבוצה אחת בלבד בליגה.
- ✓ מאמן יכול לאמן רק קבוצה אחת ולכל קבוצה יכול להיות רק מאמן אחד.

1) **Person** class המייצג בן אדם (כל האנשים אשר משחקים או עובדים בליגה):

<u>יש לשמור לכל בן אדם :</u>

- שם פרטי ושם משפחה מסוג string כל אחד.
- מספר ת.ז מסוג string.

<u>יש להגדיר למחלקה את המתודות הבאות:</u>

- בנאים
- הורס
- מתודה המדפיסה את פרטי הבן אדם.
- מתודה בוליאנית הבודקת האם הבן אדם הנבחר הוא מצטיין.(בשלב זה המתודה לא תכיל חוקית ספציפית).

למחלקה הזו ניתן להוסיף מתודות לפי הצורך.

2) **Worker** class המייצג עובד:

<u>העובד הוא גם בן אדם ויש לשמור לכל עובד בנוסף :</u>

- שנות הותק של העובד מסוג int.

<u>יש להגדיר למחלקה את המתודות :</u>

- בנאים
- הורס
- - מתודה המחזירה את המשכורת. .(בשלב זה המתודה לא תכיל חוקית ספציפית).
- מתודה המדפיסה את פרטיו של העובד, כולל הדפסת משכורתו ושנות הותק שלו.

למחלקה זו ניתן להוסיף שדות ומתודות לפי הצורך.

3) **Footballer** class המייצג שחקן:

<u>השחקן הוא גם עובד ויש לשמור לכל שחקן בנוסף:</u>

- שם הקבוצה שהוא משחק בה.
- כמות השערים שהבקיעה עד היום.
- כמות המשחקים שישחק בליגה (משתנה קבוע – מוגדר להיות 8)
- את התפקיד של השחקן בקבוצה מתוך רשימת התפקידים (Goalkeeper, Defender, Midfielder, Forward, Back)

<u>יש להגדיר למחלקה את המתודות :</u>

- בנאים
- הורס

- מתודה המחזירה את אחוזי ההצלחה שלו בליגה. (ההצלחה נמדדת כיחס בין מספר השערים שהשחקן הבקיעה למספר המשחקים ששיחק)
- מתודה בוליאנית הבודקת האם השחקן מצטיין . שחקן מצטיין אם אחוזי ההצלחה שלו עולים על 75%.
- מתודה שמגדילה את מספר השערים במידה והשחקן הבקיע שער.
- מתודה המחזירה את המשכורת . נסמן ב-x את אחוזי ההצלחה שלו. משכורתו תיהיה basis+(x/100)*2000
- מתודה המדפיסה את פרטיו של השחקן, כולל את כמות השערים שהבקיעה, את אחוזי ההצלחה שלו ואת המשכורת שלו.
למחלקה הזו ניתן להוסיף מתודות לפי הצורך.

4) **Coach** class המייצג מאמן.
<u>המאמן הוא גם עובד ויש לשמור לכל מאמן בנוסף:</u>
- שם הקבוצה שהוא מאמן.
- האם המאמן הוא שחקן עבר מסוג bool.
- מספר האליפויות שהוא לקח עם הקבוצות שאימן.

<u>יש להגדיר למחלקה את המתודות :</u>
- בנאים
- הורס
- מתודה המחזירה את המשכורת . נסמן ב-x את שנות הוותק וב-y את מספר האליפויות שלקח בהתאמה משכורתו תהיה שווה ל- basis+200*x+500*y
- מתודה בוליאנית הבודקת האם המאמן מצטיין. מאמן מוגדר כמצטיין אם הוביל את הקבוצה לפחות פעמיים לאליפות.
- מתודה המדפיסה את פרטי המאמן, כולל הדפסת המשכורת ואת שם הקבוצה שהוא מאמן.
למחלקה זו ניתן להוסיף מתודות לפי הצורך.

5) **Referee** class המייצג שופט.
<u>השופט הוא גם עובד ויש לשמור לכל שופט בנוסף:</u>
- מספר טעויות שיפוט.
-מספר משחקים שבהם שפט.

<u>יש להגדיר למחלקה את המתודות :</u>
- בנאים
- הורס
- מתודה המחזירה את המשכורת. משכורתו של שופט היא המשכורת הבסיס של עובד + 30 ש"ח עבור כל משחק שהוא שפט בו. אם השופט בעל מספר טעויות השיפוט המקסימלי המשכורת שלו יורד ב500 ש"ח.
- מתודה בוליאנית הבודקת האם השופט מצטיין. השופט מצטיין אם מספר טעויות השיפוט שלו הוא קטן בחצי ממספר הטעויות השיפוט המקסימלי . (מספר הטעויות השיפוט המקסימלי נקבע כמספר טעויות השיפוט הגדול ביותר שיש לאחד מהשופטים)
- מתודה שמקבלת את מספר טעויות השיפוט הנוספות שהשופט עשה ומעדכן בהתאם את המשתנה המתאים.
- מתודה המדפיסה את פרטיו של השופט, כולל הדפסת המשכורת והוותק ואת מספר טעויות השיפוט שלו.
- מתודה שמעדכנת את מספר המשחקים שהשופט ישפוט בהם- בכל פעם שהשופט הנ"ל משתייך לקבוצה מסויימת המתודה תגדיל ב1 את מספר המשחקים.
למחלקה זו ניתן להוסיף מתודות לפי הצורך.

6) **Manager** class המייצג מנהל הקבוצה.
המנהל הוא גם עובד **אולם** אין לשמור בו מידע נוסף (מעבר לעובד).

יש להגדיר למחלקה את המתודות :
- בנאים
- הורס
- מתודה המחזירה את משכורתו של המנהל. נסמן ב-y את שנות הוותק כך שמשכורתו תהיה שווה ל: basis*3+y*500.
- מתודה בוליאנית הבודקת האם המנהל מצטיין. מנהל מוגדר כמצטיין אם מספר שנות הוותק שלו גדול מ-4.

למחלקה זו ניתן להוסיף מתודות לפי הצורך.

7) **Stadium** class המייצג אצטדיון.
יש לשמור לכל אצטדיון:
- שם
- עיר שבה הוא נמצא
- מספר המושבים

יש להגדיר למחלקה את המתודות :
- בנאים
- הורס
- מתודה שמדפיסה את פרטי האצטדיון.

למחלקה זו ניתן להוסיף מתודות לפי הצורך.

8) **Team** class המייצג קבוצה.
יש לשמור לכל קבוצה:
-שם הקבוצה
- מצביע לאצטדיון שהיא שייכת אליו. כלומר stadium*
- מערך המצביעים לשחקנים השייכים לקבוצה , כלומר Footballer**
- כמות השחקנים ששייכים לקבוצה
-מצביע למאמן Coach*
-מצביע למנהל Manager*
- משתנה בוליאני האם קיים מנהל לקבוצה (כאשר אובייקט-קבוצה נוצר, משתנה זה יאותחל לfalse).

יש להגדיר למחלקה את המתודות :
- בנאים
- הורס
- מתודה שמוסיפה שחקן- המתודה מקבלת מצביע לשחקן ומוסיפה אותו למערך השחקנים.
- מתודה שמוסיפה מנהל – המתודה מקבלת מצביע למנהל במידה ולא קיים מנהל היא מוסיפה אותו ומעדכנת את המשתנה הבוליאני . אחרת, מדפיסה הודעה מתאימה.
- מתודה הוספת מאמן- המתודה מקבלת מצביע למאמן במידה ולא קיים מאמן היא מעדכנת אותו אחרת מציגה הודעה מתאימה.
- מתודה המקבלת מספר שלם חיובי ומחזירה את מצביע לשחקן הממוקם במיקום זו , לדוגמה מקבלת 0 ומחזירה את המצביע לשחקן הראשון המופיע במערך השחקנים בקבוצה זו.
- מתודה שמדפיסה את פרטי הקבוצה: שם הקבוצה, פרטים על אצטדיון הבית, פרטים על השחקנים, המאמן והמנהל.

למחלקה זו ניתן להוסיף מתודות לפי הצורך.

9) **Game** class המייצג משחק.
יש לשמור לכל משחק:
- מצביע לתאריך (מסוג מחלקת Date – מחלקה שבניתם במעבדה השנייה)
- מצביע לכל אחת מהקבוצות שמשחקות.
- תוצאת המשחק - שם הקבוצה המנצחת
- מספר הגולים שכל אחת מהקבוצות צברה
- רשימת השופטים (כלומר **Referee)
יש להגדיר למחלקה את המתודות :
- בנאים
- הורס
- מתודה שמעדכנת גול. המתודה מקבלת ID ובודקת האם הוא שייך לאחד מהשחקנים שמשחקים במשחק ובהתאם לקבוצה שבה השחקן הזה משחק היא מעדכנת את מספר הגולים ובמידת הצורך מעדכנת את הקבוצה המנצחת.
- הדפסת פרטי המשחק. הקבוצות המתחרות ואת פרטי המאמן שלה. והדפסת הקבוצה המנצחת.
למחלקה זו ניתן להוסיף מתודות לפי הצורך.

10) **Ligat_HaAl** class המייצגת את ליגת העל.
יש לשמור במחלקה הזו:
- מערך דינאמי של מצביעים לכל הקבוצות בליגה, כלומר **Team
- מערך דינאמי של כל המשחקים בליגה , כלומר **Game
- מערך דינאמי של כל העובדים בליגה , כלומר **Worker**
- כמות העובדים בליגה
* הערה : אין צורך ביצירת אותו אובייקט פעמים (כלומר, לרשימת העובדים בקבוצה (Teamב) ניתן להוסיף ולשמור את הכתובת המקורית ולא ליצור העתק).
יש להגדיר למחלקה את המתודות :
-בנאי ברירת המחדל :
1. יוצרים 5 קבוצות כאשר בכל קבוצה יש בשלב ההתחלתי 2 שוערים, 3 חלוצים, 4 קשרים, 3 מגנים, 3 בלמים , מאמן ומנהל.
2. מאתחל את מערך השופטים – בשלב ההתחלתי יש 4 שופטים בליגה.
3. בונה את לוח המשחקים, שימו לב שכל קבוצה משחקת נגד כל אחת מהקבוצות השניות בליגה פעמיים (על פי ההוראות שניתנו למעלה), תאריך המשחק והשופטים יוגרל בצורה דינאמית בזמן ריצה .
* לא לאתחל את המשתמשים של המחלקה בהכרזה אלא רק ע"י בנאים
-הורס
- מתודה **Menu** אשר תדפיס למשתמש את כל האופציות הנתונות בפניו בתפעול המערכת ותיתן לו את היכולת לבחור איזו אפשרות ברצונו להפעיל. המתודה תיתן למשתמש את האפשרויות הבאות:
1 ***Add Footballer*** **-** הוספת שחקן. על המשתמש להקליד את הקבוצה שאליו השחקן מצטרף ופרטי השחקן (שם, שם משפחה, ת.ז, ותק, ותפקידו). על המערכת להוסיף את השחקן לקבוצה ולמערך כללי של העובדים. במידה ובן אדם עם אותו ת.ז. כבר קיים במערך העובדים יש להדפיס הודעה מתאימה ולא להוסיף אותו.
2 - ***Add Referee*** - הוספת שופט לליגה. על המשתמש להקליד את פרטי השופט כולל שנות וותק. על המערכת להוסיף את השופט למערך הכללי הליגה. אם בן אדם עם אותו ת.ז. כבר קיים יש להדפיס הודעה מתאימה ואין להוסיף אותו.

3 - ***Print Team details***- הדפסת פרטי קבוצה לפי שם הקבוצה. על המשתמש יש להקליד את מספר הקבוצה שהוא רוצה לראות את הפרטים שלה (אחרי שהוצגו בפניו האפשרויות).
4 - ***Print referee details*** - הדפסת כל השופטים בליגה. יש להדפיס את הפרטים של כל השופטים בליגה, כולל את המשחקים שהוא שפט בהם.
5 - ***Print future games details*** - הדפסת כל המשחקים העתידים להתקיים. יש להדפיס את הפרטים על המשחקים שמתקיים אחרי התאריך של יום הבדיקה. יש להדפיס את שמות הקבוצות המשחקות ואת התאריך שהמשחק יתקיים בו.
6- ***Print worker salary*** – הדפסת משכורת של עובד ע"י הקלדת הת.ז שלו. במידה והת.ז שהוקלד לא תקין תוצג הודעה מתאימה.
7- ***Add Goal*** – הוספת גול למשחק. על המשתמש לבחור משחק (מתוך רשימת האופציות לבחירה שבו יופיע מספר משחק ואת שמות הקבוצות המשחקות ותאריך) שבו הוא רוצה לעדכן על גול נוסף. אחרי שהמשתמש בחר את המשחק הוא יקליד את ת.ז של השחקן שהבקיע גול. במידה ושחקן זה קיים באחת מהקבוצות המשחקות במשחק הנבחר נעדכן את מספר הגולים שהבקיע השחקן ואת תוצאת המשחק.
8- ***Add judgment errors*** – הוספת טעויות שיפוט. על המשתמש להקליד ת.ז של שופט שהוא רוצה להוסיף לו טעויות שיפוט חדשות. במידה והת.ז שהוקלד לא שייך לשופט תוצג הודעה מתאימה. אחרת, נעדכן את מספר טעויות השיפוט של שופט הנ"ל.
9 - ***Print outstanding people*** - הדפסת מצטיינים. יש להדפיס את כל הפרטים של כל <u>האנשים המצטיינים</u> אשר עובדים בליגה (שחקנים/שופטים/מאמנים/מנהלים). עבור שחקנים יש להדפיס גם את אחוזי ההצלחה, עבור מאמנים יש להדפיס אם הם שחקני עבר ואת מספר האליפויות, ועבור שופטים את מספר טעויות השיפוט שלהם והאם הם בעלי מספר טעויות שיפוט מקסימלי.
10 - ***Print table*** – הדפסת טבלת ניקוד. הטבלה תכלול את שם הקבוצה ואת הניקוד. הניקוד נקבע ע"י כך שניצחון משחק מקנה 3 נקודות נוספות לקבוצה המנצחת ותיקו מקנה נקודה אחת לשתי הקבוצות ששיחקו. במידה ועדין לא הוקלדו תוצאות המשחק הניקוד שינתן עבור משחק זה יהיה 0.
11 - ***Print footballer details with smallest salary*** - הדפסת פרטי שחקן כדורגל בעל המשכורת המינימלית (יש להדפיס את כל המידע הרלוונטי).
12 –***Exit*** - יציאה מהמערכת.

למחלקה הזו ניתן להוסיף מתודות אך הן חייבות להיות פרטיות למחלקה (מתודות get הכרחיות יכולות להיות ציבוריות).
להלן הפונקציה הראשית (main):

```
#include "Ligat_HaAl.h"
int main(){
        Ligat_HaAl liga;
        liga.Menu();
  return 0;
}
```