תרגיל 3 - הנחיות

בתרגיל 3 עליכם להגיש:

1. תכנית שיודעת להריץ קרבות בין שחקנים לפי ההנחיות שיפורטו לעיל (מתקמפלת לקובץ הרצה)
2. אלגוריתם שחקן שידע להתקמפל לקובץ so ושיתחרה מול חבריכם
3. קובץ makefile שידע לייצר בפקודה אחת את 1 ו-2 לעיל (הנחיות מדוייקות יפורסמו בהמשך)

תכנית להרצת קרבות בין שחקנים

התכנית תדע לקבל ב-command-line שני פרמטרים (אופציונליים) בסדר כלשהו:

-threads <num_threads>

-path <location_of_algorithms>

הפרמטר של מספר ה-threads ייקבע עבור מנהל התחרות כמה threads לפתוח בו זמנית לצורך הרצת תחרות. אין חובה להשתמש בכל ה-threads אם אין צורך (למשל אם יש רק שני אלגוריתמים מתחרים) - אבל אסור לפתוח יותר. ה-thread הראשי נכלל בספירה - כלומר אם המספר שניתן הוא 1 המשמעות היא שאין לפתוח threads נוספים.

במידה ולא נמסר פרמטר threads ברירת המחדל היא 4 threads (כולל ה-thread הראשי כאמור).

הפרמטר של ה-path קובע מהיכן לטעון קבצי so לתחרות. במידה והפרמטר לא נמסר התכנית תחפש את קבצי ה-so ב-working directory.

אופן הרצת התחרות

התכנית צריכה לטעון את כל קבצי ה-so בהתאם ל-path (בין אם ניתן או שנעשה שימוש בברירת המחדל).

במידה ואין בכלל קבצי so (או שיש רק אחד) יש להדפיס הודעת usage מתאימה והגיונית לבחירתכם שתסביר את הבעיה.

האלגוריתמים יירשמו באופן אוטומטי כחלק מטעינת ה-so - כפי שיוסבר להלן.

מנהל הטורניר ידאג להריץ כל אלגוריתם 30 משחקים מול אלגוריתם אחר כלשהו - ובלבד שכל אלגוריתם לא ישחק נגד עצמו ולא ישחק יותר מ-30 משחקים אלא בדיוק 30 משחקים. מותר לכם לייצר טורניר באופן רנדומלי (להגריל יריבים) או לדאוג ש-30 המשחקים יתפלגו באופן יחסית אחיד בין כולם ככל שאפשר. במידה ונוצר מצב,

מסיבה כלשהי, שחייבים להריץ משחק בין אלגוריתם שעדיין לא שיחק 30 משחקים לבין כזה שכבר שיחק 30 משחקים - יש לצבור נקודות רק לזה שלא שיחק.

שימו לב: מנהל הטורניר יכול לייצר שני instances של אותו אלגוריתם לצורך משחק במקביל, על-מנת לאפשר זאת עליכם לוודא שאתם לא משתמשים במשתנים סטטיים באלגוריתם המשחק שלכם.

צבירת נקודות

אלגוריתם שניצח משחק מקבל 3 נקודות.
במצב של תיקו שני הצדדים זוכים לנקודה.

הדפסת תוצאות הריצה

בסיום כל הקרבות יש להדפיס למסך את רשימת השמות של האלגוריתם בסדר יורד לפי מספר הנקודות שצברו וליד כל אלגוריתם את מספר הנקודות (עם רווח בין השם לבין מספר הנקודות).
דוגמה:

072314249 90
345242483 45
907231424 0

אלגוריתם שחקן

האלגוריתם שאתם מגישים אמור להתקמפל לקובץ so שיקבל את השם:

RSPPlayer_<ID>.so

כאשר את <ID> יחליף ה-ID של אחד מהמגישים (לא משנה מי) - בלי סוגריים משולשים!
למשל:

RSPPlayer_345242483.so

שם המחלקה

כדי לבחור שם מחלקה לאלגוריתם השחקן שלכם, שלא יתנגש עם מחלקות של חבריכם - ומבלי צורך לחייב שימוש ב-namespaces, שם המחלקה שתבחרו חייב להיות זהה לשם קובץ ה-so, למשל:

```
class RSPPlayer_345242483 : public PlayerAlgorithm { ... };
```

רישום אוטומטי של המחלקה

על-מנת שמחלקת המשחק שלכם תירשם אוטומטית ברגע שה-so נטען, עליכם לשבץ בקובץ ה-cpp של מחלקת האלגוריתם שלכם, הכרזה גלובלית כזו:

REGISTER_ALGORITHM (<ID>)

למשל:

REGISTER_ALGORITHM (345242483)

השורה הזו תגרום לרישום של האלגוריתם בהתבסס על ה-header המשותף "AlgorithmRegistration.h" שיפורסם ליד קובץ התרגיל.

<u>שימו לב:</u> האחריות למימוש AlgorithmRegistration.cpp היא עליכם, כחלק מהמימוש של מנהל המשחק שלכם.

עד כמה האלגוריתם שלנו צריך להיות חכם?

אין חובה לשפר את האלגוריתם שלכם מתרגיל 2 - אם הוא היה חכם בצורה סבירה כלשהי (כלומר, לא היה לגמרי טיפש).

ניתן לקבל 100 גם אם תהיו אחרונים בתחרות (בהנחה שהתרגיל שלכם מושלם משאר הבחינות).

<u>אבל</u> - המקומות הראשונים בתחרות יזכו לכבוד ויקר ולבונוס מכובד שישפיע על הציון הסופי (ישוקלל בחישוב הציון הסופי, גם אם ציון התרגיל יעבור את ה-100).

לכן כדאי להשקיע. אבל - לא חובה.

אז מה בעצם העיקר בתרגיל?

- להגיש אלגוריתם שיודע להיבנות ל-so, לרשום את עצמו אוטומטית ולרוץ בתחרות.
- להגיש מנהל טורניר שמייצר טורניר כמו שהוגדר. כולל מימוש של AlgorithmRegistration משלכם שתואם ל-header שפורסם.
- להשתמש במנהל הטורניר ב-threads באופן נכון.

בהצלחה!