Задатак 1, Хиподром

У конкурентном окружењу симулирати систем коњске трке на хиподрому. На располагању би требало бити 10 различитих коња за сваку трку са својим јединственим именом. Задавање улазних вриједности за трку (датум трке, имена коња) треба омогућити прослеђивањем улазне датотеке са релевантним информацијама. Свака трка треба да има засебну улазну датотеку за конфигурацију које ће се учитавти редом док програм не испарсира све улазне датотеке које су везане за наведену конфигурацију (нпр. Све датотеке из одређеног директоријума). Хиподром би требао имати 10 трака дужине 1000 мјеста у којој учествује 10 тркаћих коња.
Сваког учесника трке треба реализовати преко засебне нити. Постоји могућност одустајања из трке која треба да буде 10% по учеснику. Учешће у трци треба еваулирати прије почетка сваке трке за сваког учесника. У случају да буде више од 30% изостанака са трке она се отказује и одговарајући извјештај треба да се направи на терминалском прозору као и посебној датотеци везаној за ту трку.
Сваки коњ треба да има физичку спремност (кондиција) која треба да се креће од 80 до 100% и која треба да се иницијализује прије сваке трке.
Кретање коња се рачуна на следећи начин:

*корак= (кондиција * насумични_број)*

и заокружује се на најближу цјелобројну вриједност. Насумичан број треба да буде између 7 и 10. Наконг сваког завреног кретања играча/коња потребно је успавати нит која је вршила кретање на одређено вријеме (нпр. 300ms или више, водити се логичним закључивањем у односу на испис/логовање трке).
Свака промјена коња треба бити записана у лог датотеку за дату трку која ће имати јединствено име (нпр. базирано на датуму трке). Уписивање статуса сваког засебног коња у датотеке не смије утицати на резултате трке.
Потребно је омогућити очитавање тренутног стања трке у сваком тренутку, док само очитавање не смије утицати на резултат трке.
Резултате трке уписивати у кружни бафер који ће бити дијељен са другом нити која такође резултате са трке аутомобила уписује резултате у тај исит бафер, док 3. нит чита резултате и празни кружни бафер. У случају да је заједнички бафер попуњен обезбједити механизам синхронизације који ће омогућити упис у заједнички кружни бафер у тренутку када он буде имао слободне позиције док сам програм треба да настави са парсирањем следеће улазне датотеке и започиње се следећа трка.

Потребно је раздвојити интерфејсе од импелентације.

Омогућити рад апликације са информацијама за дебаговање и без њих.

Програм се прекида уносом слова „q“ са тастатуре.

Задатак 2, Трка аутомобила

У конкурентном окружењу симулирати систем тркаће стазе за аутомобиле. На располагању би требало бити 9 различитих аутомобила за сваку трку са својим јединственим именом. Задавање улазних вриједности за трку (датум трке, број кругова, стартна мјеста, имена аутомобила) треба омогућити просљеђивањем улазне датотеке са релевантним информацијама. Свака трка треба да има засебну улазну датотеку за конфигурацију које ће се учитавти редом док програм не испарсира све улазне датотеке које су везане за наведену конфигурацију (нпр. Све датотеке из одређеног директоријума). Тркаћа стаза треба да има 3 траке и дужину од 100 поља у којој учествује 9 тркаћих аутомобила.
Сваког учесника трке треба реализовати преко засебне нити. Могу да постоје 2 временска услова на стази, мокро и суво, који се додјелјују насумично стази прије почетка трке.
Сваки тркаћи аутомобил треба да посједује 2 сета гума, гуме за суво и за мокро вријеме, које ће се такође додјелити приликом иницијализације аутомобила. Кретање аутомобила напријед треба бити дефинисано насумичним генратором бројева између 7 и 10 када је суво вријеме, а 3 и 6 када је стаза влажна. Број добијен из наведених генератора се евалуира у зависности од типа гума које посједује аутомобил како би се добио укупан број поља које прелази аутомобил на следећи начин:

Временски услови на стази	Тип гума	Генератор	Корак
суво	суво	број	број
мокро	суво	број	број*0,7
суво	мокро	број	број*0,8
мокро	мокро	број	број

Добијени број треба заокружити на најближу цјелобројну вриједност.
У зависности од тога у којој се траци налази аутомобил заобилажење аутомобила испред себе је дефинисано на начин да ће аутомобил морати да се престрои у другу или трећу траку лијево или десно док не нађе слободну траку за кретање унапријед. При сваком престројавању аутомобил троши број поља у зависности од броја престројавања који је добио претходном калкулацијом.
Свака промјена аутомобила треба бити записана у лог датотеку за дату трку која ће имати јединствено име (нпр. базирано на датуму трке). Уписивање статуса сваког засебног аутомобила у датотеке не смије утицати на резултате трке. Наконг сваког завреног кретања играча/аутомобила потребно је успавати нит која је вршила кретање на одређено вријеме (нпр. 300ms или више, водити се логичним закључивањем у односу на испис/логовање трке).
Потребно је омогућити очитавање тренутног стања трке у сваком тренутку, док само очитавање не смије утицати на резултат трке.
Резултате трке уписивати у кружни бафер који ће бити дијељен са другом нити која такође уписује резултате са хиподрома и уписује их у тај исит бафер, док 3. нит чита резултате и празни кружни бафер. У случају да је заједнички бафер попуњен обезбједити механизам синхронизације који ће омогућити упис у заједнички крућни бафер у тренутку када он буде имао слободне позиције док сам програм треба да настави са парсирањем следеће улазне датотеке и започиње се следећа трка.
Потребно је раздвојити интерфејсе од импелентације.
Омогућити рад апликације са информацијама за дебаговање и без њих.
Програм се прекида уносом слова „q“ са тастатуре.

Задатак 3, Kladionica

У конкурентом окружењу симулирати систем кладионице који има 3 метода рада. Метод Live Score, метод Results и метод Race. Избор метода рада реализовати уносом неког знака са тастатуре (нпр. броја 1, 2 ... или слова „l“, „r“).

Метод Live Score реализовати преко нити. Омогућити приказ свих тренутних резултата „трка“ које нису завршене (релевантне информације) у терминалском прозору. Приказ мора бити реализован на читак и прегледан начин, поредан по неком логичном редослиједу. Одговарајући тренутни резултати се читају директно из доступних трка („са терена“) на начин који неће утицати на резултате саме трке. Приказ мора бити редовно освјежен са најновијим резултатима.
Метод Results реализовати преко нити. Омогућити приказ резултата „трка“ које су завршене (релевантне информације) у терминалском прозору. Приказ мора бити реализован на читак и прегледан начин, поредан по неком логичном редослиједу. Одговарајући тренутни резултати требају бити прочитани из заједничког кружног бафера и након читања смејштени у меморију кладионице. Када одређени резултати постану застарјели, они се смјештају у заједничку датоткеу у којој требају бити смјештени сви резултати респективно. На крају извршавања читавог програма ова датотека треба да садржи све релевантне информације о свим завршеним тркама.
Метод Race треба да има више модова рада у зависности од броја трка. Када се унесе одговарајући мод рада (нпр. 3, за метод Race па онда 1 за прву трку) треба да приказује тренутни статус кокретне трке са активним приказом на екрану који ће се мијењати са активним промјенама на тркаћим стазама.

Пр.

```
-------------------------
****** Trka konja ******
-------------------------
Datum: 19.5.2017
-------------------------
Ucesnici:
s1: Konj1
s2: Konj2
s3: Konj3
s4: Konj4
s5: Konj5
s6: Konj6
s7: Konj7
s8: Konj8
s9: Konj9
s10: Konj10


************
     3
************
     2
************
     1
************
  *START*
======================
s1: ------*------------------
s2: --------*----------------
s3: --------*----------------
s4: -------*-----------------
s5: ---------*---------------
s6: -------*-----------------
s7: ------*------------------
s8: -------*-----------------
s9: ------*------------------
s10:----------*--------------
```

Слично треба и за трку аутомобила (Prikaz treba da prati prvog (vodeceg) trkaca +- 20 pozicija ispred i iza).

Потребно је раздвојити интерфејсе од импелентације.

Омогућити рад апликације са информацијама за дебаговање и без њих.

Програм се прекида уносом слова „q“ са тастатуре.

Тим који буде радио 3 задатак, покрај дијела који је дефинисан у само тексту задатка мора да уради интеграцију имплементације задатка 1, 2 и 3 у један заједнички програм.

Заједнички дио:

Прије интреграције система у један задатак, потребно је да сви дијелови могу засебно да функционишу и да се истестирају као посебне цјелине.
Сви дијелови задатка који нису прецизно дефинисани и остављени за слободну итерпретацију програмерима који буду радили имплементацију захтјева.
За сваки дио пројектног задатка треба бити направљена одговарајућа презентација као и заједничка презентација која ће да обједини читав пројектни задатак.
У случају да један од 3 тима има одређених застоја који могу итицати на реализацију читавог пројектног задатка, дозвољена је привремена алокација одређеног броја чланова тима у други тим са адекватним образложењем.
Поштиовати правила кодовања која су достављена у посебном фајлу.
Потребно је адекватно документовати свој код користећи “Doxigen” начин коментарисања за Це код као и генерисати одговарајући “HTML” помоћу наведеног алата.