NÉV:		NEPTUN KÓD:							**D** CSOPORT

!	**A beadandó <u>solution elnevezési</u> mintája: NEPTUNKOD_ D (neptunkod = az Ön saját kódja)**
	A beadott dolgozat fordítási hiba esetén <u>nem</u> értékelhető. Kommentezett részre pont <u>nem</u> szerezhető.
	A feltöltendő .zip állományt a megfelelő kurzushoz töltse fel, egyéb esetben a ZH <u>nem</u> értékelhető.

1. NAGY ZÁRTHELYI DOLGOZAT

Készítsen konzolos Pacman játéktáblát!

Szeretnénk alkalmazást készíteni, amely majd egy stabil alapot fog szolgálni a fejlesztőcsapat másik részének, akik a dinamikai részt fogják megvalósítani a játék kapcsán. A feladat alapján egy kettő dimenziós tömb fogja szolgáltatni, amelyen kérték, hogy különböző elemzéseket végezzünk. Így a meglévő vizsgálatokat már az Ön által megírt metódusok fogják szolgáltatni a fejlesztés későbbi részében. Végezzen tehát alapos munkát, és csak futtatható kódot adjon le – egyéb esetben a továbbfejlesztés nem fog működni.

1. char[,] PalyaGeneralas()

Előállít egy megfelelő méretű kétdimenziós tömböt, amely a játékteret reprezentálja és a megfelelő véletlen adatokkal feltölti. A teljes játéktér 30 oszlopból és 15 sorból áll. A pályát a következő szabályok alapján generálja le:

1. a pálya bal és jobb oszlopai, valamint felső és alsó sorai a pálya szélét reprezentálják, ide kerüljön 'O' karakter
2. a pálya többi részén véletlenszerűen döntse el az algoritmus, hogy további fal található ott 'O', vagy felvehető coin 'C', vagy egyszerű háttér '-' található. Figyeljen arra, hogy döntő többségben egyszerű háttér legyen, fal és coin csak kis százalékban forduljon elő, például 20% esély legyen a coinra és falra, 80% pedig a sima háttérre.
3. a pálya egy véletlenszerűen előállított pontjára helyezze el Pacman-t 'P', amely majd az irányított karakter lesz. Annyi megkötés van, hogy a pálya széleihez nem kerülhet, tehát a felső, alsó valamint jobb és bal sorokat / oszlopokat hagyja ki.
4. alkalmazza a 3. feladatban található ellenség generálására szolgáló metódust, az ott leírtaknak megfelelően.

2. void Megjelenit(char[,] palya)

A bemeneti mátrixot kiírja a konzolra, a megfelelő sor és oszlop formátumban. Az oszlopok értékei space-szel legyenek elszeparálva. Az irányítandó karaktert piros színnel rajzolja ki. A felvehető coin-okat kék színnel, az ellenségeket pedig magenta színnel.

– folytatás a túloldalon –

3. void EllensegGeneral(char[,] palya)

A metódus generáljon le 30 db ellenséget, amelyet véletlenszerűen helyezzen el a játéktéren. Fontos, hogy az irányítandó karaktert (Pacman-t, 'P') ne írja felül, így, ha azzal megegyező cellába kerülne az ellenség, generáljon új helyet az ellenségnek. Folytassa a generálást mindaddig, amíg jó helyre nem kerül. Jelölje 'E' az ellenségeket.

4. char MezoLekerdezes(char[,] palya)

Bekér a felhasználótól a konzolról egy karaktersorozatot a következő formában ''<A@B>'', amelyből az A és B értékek fogják meghatározni a játéktéren keresett mezőt. A jelentse a sorszámot, B pedig az oszlopszámot. Az itteni mezőn található mezőelemet adjuk vissza (pl. P mint Pacman, C mint coin stb.). A feladatban a < és > jelek is számítanak!

5. double[] CoinAranyszamSoronkent(char[,] palya)

Tömb formájában visszaadja, hogy egyes sorokban mennyi a coin-ok arányszáma. Arányszám alatt a coin-ok darabszáma / összes mező az adott sorban hányadost értjük.

6. double[] MaxCoinDarabszam(double[] coinAranySoronkent)

Az 5. feladat metódusából előállt adathalmazt megvizsgálja, és visszaadja annak a sornak az indexét és értékét, ahol a coin-ok darabszáma a legnagyobb volt. A visszaadáskor egy két elemű tömböt alkalmazzon, első helyre az index, második helyre pedig az érték kerüljön.

7. int[] EllensegOszloponkent(char[,] palya)

A metódus a pályát bejárva meghatározza, hogy azon oszloponként hány darab ellenség található, ennek eredményét pedig tömbben visszaadja.

8. int[] MaxEllensegszam(int[] ellensegOszloponkent)

Határozza meg, hogy mely oszlopokban volt a legtöbb az ellenség.

9. void RendezesCsokkenobe(double[] coinAranySoronkent)

Az 5. feladat metódusából előállt adathalmazt kapja meg bemeneti tömbként, és annak az elemeit csökkenő sorrendbe rendezi.

A Main() függvényből tesztelje le mindegyik elkészített metódus működését!

Minden részfeladat elvégzését a megadott, külön metódussal valósítsa meg!

Az egyes metódusok megvalósítása során – ahol lehetséges – a tanult programozási tételeket és rendezési algoritmusokat használja!

ZÁRTHELYI DOLGOZAT BEADÁSA

A kész feladatot **teljes egészében** *(egész solution mappa)*, .zip (nem .rar, .7z stb.) formába tömörítse be, majd a *http://zh.nik.lan* leadó felületen, a **megfelelő kurzust kiválasztva** töltse fel!

Megfelelő kurzus a Szoftvertervezés és –Fejlesztés I. (nappali / NIXSF1HBNE), azon belül pedig az az alkalom, amelyre Ön jár. Rossz kurzusra való feltöltés esetén a zárthelyi nem értékelhető. Fordítási hibával rendelkező program nem érétkelhető.