# Strukture podataka i algoritmi

## Projektni zadatak 002\_POKEMON

Vaš zadatak je napisati konzolni program u C++ koji će korisniku omogućiti rad s Pokemonima. Program mora biti smisleno strukturiran i intuitivan. U nastavku su opisane funkcionalnosti koje treba program imati. Za izradu projekta se koristi datoteka **SPA\_PROJ\_002\_POKEMON\_data.csv** čiji je opis sljedeći (ako neki podatak nedostaje, zamijenite ga nulom ili praznim stringom):

* #: ID for each pokemon
* Name: Name of each pokemon
* Type 1: Each pokemon has a type, this determines weakness/resistance to attacks
* Type 2: Some pokemon are of dual type (Type 1 and Type 2). Type 1 is obligatory, and Type 2 is optional
* Total: sum of all stats that come after this, a general guide to how strong a pokemon is
* HP: hit points, or health, defines how much damage a pokemon can withstand before fainting
* Attack: the base modifier for normal attacks (eg. Scratch, Punch)
* Defense: the base damage resistance against normal attacks
* SP Atk: special attack, the base modifier for special attacks (e.g. fire blast, bubble beam)
* SP Def: the base damage resistance against special attacks
* Speed: determines which pokemon attacks first each round
* Generation: number of generation
* Legendary: is pokemon legendary or not

1. (Bodova: **1**) Pokretanjem aplikacije prikažite korisniku sljedeći izbornik:

Pretraga po ID

Prikaz po snazi napada

Kopiranje u novi vektor

Prikaz po generacijama

1. (Bodova: **2**) Dizajnirajte i implementirajte tip podataka Pokemon koji može čuvati sve podatke o pokemonima iz datoteke. Pripremite tip podataka tako da se može konstruirati na smislene načine te definirajte gettere, settere i potrebne metode za svakog člana. Omogućite korisniku da pozivanjem metode na objektu može ispisati osnovne podatke o pokemonu na ekran (naziv, tip 1, tip 2, total).
2. (Bodova: **2**) Kad korisnik odabere „Pretraga po ID“, prvo prekopirajte pokemone iz vektora u odgovarajući kontejner (ID-evi nisu jedinstveni u datoteci). Zatim pitajte korisnika da upiše ID kojeg želi tražiti pa mu nakon toga ispišite podatke o svim pokemonima s tim ID-em ili napišite da ne postoji pokemon s tim ID-em. Osigurajte da se pretraživanje odvija u logaritamskoj složenosti koristeći najbolji od sljedećih kontejnera: map, multimap, set, multiset.
3. (Bodova: **2**) Promijenite prethodni zadatak tako da pretraživanje radite prvo po odabranom kontejneru, a zatim i po vektoru i po listi (kopirajte podatke i u listu). Ispišite top ljestvicu brzine kontejnera: prvo prikažite najbrži, onda srednji pa najsporiji.
4. (Bodova: **2**) Kad korisnik odabere „Prikaz po snazi napada“, pitajte korisnika da odabere prikaz od slabijih prema jačim ili od jačih prema slabijim pokemonima (svojstvo „Attack“). Koristeći prioritetni red, ispišite korisniku sve pokemone, poštujući odabrani redoslijed.
5. (Bodova: **3**) Kad korisnik odabere „Kopiranje u novi vektor“, kreirajte novi vektor *v* i napravite sljedeće:

* Prekopirajte sve pokemone u novi kontejner koji omogućuje da se pretraživanje po svojstvu „Type 1“ odvija u logaritamskoj složenosti (optimalno birajte map, multimap, set ili multiset).
* Sve dok to korisnik želi, omogućite mu unos vrijednosti za „Type 1“ (Grass, Fire, Water, …).
* Kad korisnik unese vrijednost, pronađite sve pokemone tog tipa u vašem odabranom kontejneru i prekopirajte ih u vektor *v*.
* Na kraju ispišite pokemone iz vektora *v*.

1. (Bodova: **3**) Kad korisnik odabere „Prikaz po generacijama“, koristeći prioritetni red ispišite sve pokemone po generacijama, počevši od prve. Pokemone iz iste generacija obavezno prikažite abecedno prema imenu.
2. (Bodova: **1**) Dodajte u izbornik sljedeće opcije:

Selection vs Insertion

Pogodi trajanje

Lets Bogo!

The Merge Dance

1. (Bodova: **3**) Kad korisnik odabere „Selection vs Insertion“, napravite sljedeće:

* Kreirajte dva pomoćna vektora *v1* i *v2* i iskopirajte u svako od njih sve pokemone.
* Uzmite s predavanja Selection i Insertion sortiranja te ih promijenite tako da sortiraju pokemone, rastuće po imenu.
* Sortirajte vektor *v1* koristeći Selection sort.
* Sortirajte vektor *v2* koristeći Insertion sort.
* Ispišite koliko je trajalo svako sortiranje.

1. (Bodova: **2**) Promijenite prethodni zadatak tako da svaki algoritam pri sortiranju zbraja broj zamjena mjesta (*swap*) koje napravi i vraća to kao izlaznu vrijednost iz funkcije. Sortirajte pa ispišite za oba sortiranja broj zamjena mjesta.
2. (Bodova: **3**) Kad korisnik odabere opciju „Pogodi trajanje“, generirajte jedan slučajni broj *n* između 1 i 100.000.000 te pitajte korisnika da upiše koliko milisekundi misli da će trajati razbacivanje, sortiranje i binarno pretraživanje tog polja u potrazi za brojem 7. Kad korisnik upiše vrijednost, generirajte to polje s vrijednostima od 1 do *n*, razbacajte ga, sortirajte ga te binarnim pretraživanjem i pronađite broj 7. Ispišite korisniku kolika je razlika između njegove pogođene vrijednosti trajanja i konkretnog trajanja.
3. (Bodova: **4**) Kad korisnik odabere „Lets Bogo!“, slučajnim odabirom prekopirajte pet pokemona u novi vektor. Isprogramirajte svoju vlastitu determinističku verziju bogo sorta te je iskoristite za slaganje pokemona po imenu u rastućem redoslijedu. Ispišite na ekranu svaku isprobanu permutaciju.
4. (Bodova: **4**) Kad korisnik odabere opciju „The Merge Dance“, pitajte ga da upiše broj *n* koji predstavlja broj elemenata. Nakon što upiše broj, u vektor ubacite slučajnim redoslijedom *n* pokemona. Zatim vektor sortirajte Merge sortom (koristite i promijenite prema potrebi implementaciju s predavanja) te za vrijeme sortiranja radite sljedeće:

* Svaki put kad se napravi poziv rekurzivne funkcije, brojač povećajte za 1
* Svaki put kad se završi poziv rekurzivne funkcije, brojač smanjite za 1.
* Za svaku promjenu brojača iscrtajte u tekstualnu datoteku jedan redak koji sadrži onoliko znakova # kolika je trenutna vrijednost brojača.

1. (Bodova: **1**) Dodajte u izbornik sljedeće opcije:

Pretraga po Type 2

Vizualiziraj

Izrada stringa iz stringa

1. (Bodova: **2**) Kad korisnik odabere „Pretraga po Type 2“, prvo prekopirajte pokemone iz vektora u odgovarajući kontejner. Zatim pitajte korisnika da upiše „Type 2“ koje želi tražiti pa mu nakon toga ispišite podatke o svim pokemonima tog tipa. Osigurajte da se pretraživanje odvija u najboljoj mogućnoj složenosti koristeći neki od sljedećih kontejnera: unordered\_map, unordered\_multimap, unordered\_set, unordered\_multiset.
2. (Bodova: **2**) Promijenite prethodni zadatak tako da pretraživanje radite prvo po odabranom kontejneru, a zatim i po vektoru i po listi (kopirajte podatke i u listu). Ispišite top ljestvicu brzine kontejnera: prvo prikažite najbrži, onda srednji pa najsporiji.
3. (Bodova: **5**) Kad korisnik odabere „Vizualiziraj“, natrpajte sve pokemone u unordered\_multimap tako da ključ bude njihov ID. Nakon toga, vizualizirajte *hash* tablicu iz unordered\_multimap tako da iscrtate sve njene *buckete* te uz svaki *bucket* prikažete i pokemone u njemu. Primjer početka ispisa (vrijednosti su izmišljene):

Bucket 0: Rapidash -- Chansey -- Chikorita

Bucket 1: Venusaur -- VenusaurMega -- Venonat -- Cubone

Bucket 2: [EMPTY]

1. (Bodova: **6**) Kad korisnik odabere „Izrada stringa iz stringa“, omogućite mu da upiše dva stringa te mu zatim, koristeći unordered\_map, unordered\_multimap, unordered\_set ili unordered\_multiset, ispišite može li se drugi string kreirati koristeći raspoloživa slova iz prvog. Primjerice, ako je prvi string „kk“, a drugi „k“, onda se drugi može konstruirati iz prvog; ako je prvi string „kkm“, a drugi „kk“, onda se drugi također može konstruirati iz prvog; no, ako je prvi string „kk“, a drugi „kkm“, onda se drugi ne može konstruirati iz prvog jer nedostaje slovo m.