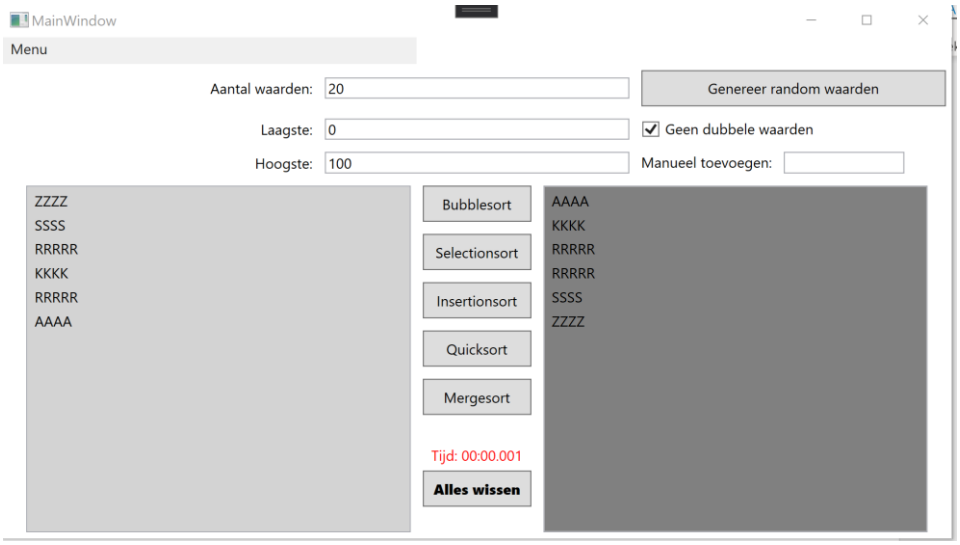
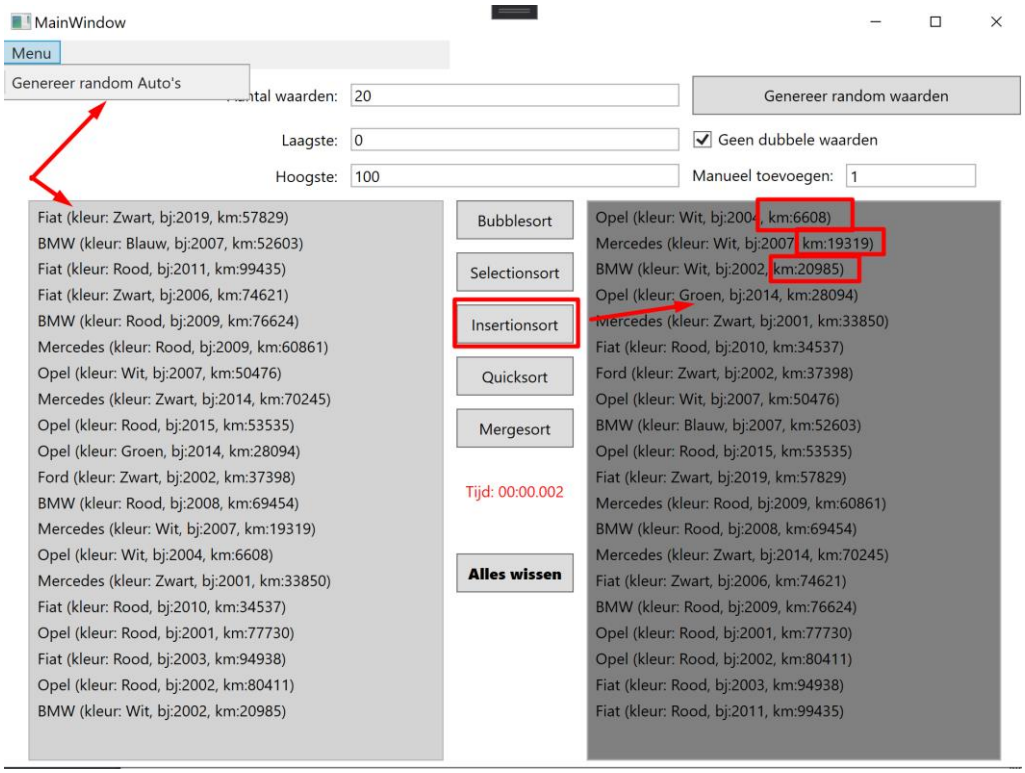
Oefeningen Zoekalgoritmen:

1. **Lineair Search**
    a. Bouw een lineair search algorithme
    b. Test deze met een lijst vanuit een console app
2. **Binary Search**
    a. Bouw een binary search algoritme op basis van recursie
    b. Test deze met een lijst vanuit een console app
3. **Hashtable**
    a. Bouw een hashtable die geen collisions kan verwerken (geeft een exceptie bij een collision)
    b. Test deze vanuit een console app
    c. Bouw een hashtable die wel collisions kan verwerken en deze afhandelt mbv. een SLL
    d. Test deze vanuit een console app

4. Selection sort
    a. **Bouw zelf** het algoritme van de Selection sort gebaseerd op de info uit de cursus en de kennis die je reeds hebt van het Bubble sort algoritme. (Uiteraard als methode in een eigen klasse.)
        i. Dwz. je weet reeds dat er **2 for lussen** nodig zullen zijn.
        ii. De **binnenste for lus** zal de array overlopen op zoek naar de minimum waarde (**compare** operaties). Hierna al zal er dan niet een **swap** operatie moeten gebeuren.
        iii. Nadien komt er een volgende iteratie, enz.. tot de lijst gesorteerd is.
    b. Test het met verschillende input.
    c. Meet tevens de performantie voor verschillende input waarden.
    d. Maak eens een 2e versie van de **selection sort** die ook **strings** kan sorteren. Zorg dat de gebruiker manueel ook strings kan invoeren zodat je dit kan testen.



5. Insertion sort
    a. Bouw vervolgens ook het insertion sort algoritme. Baseer je ook hier weer op de info uit de cursus.
    b. Test en meet de performantie (1000, 10000, 20000, 30000, 50000, 100000,… elementen)
    c. Maak een nieuwe versie van het insertion algoritme dat in staat is om Auto's te sorteren op basis van hun km stand. (dus voorzie zelf een klasse Auto met deze property en nog enkele andere naar keuze: bouwjaar, model, kleur,..)



6. Quick Sort
    a. Ga op zoek naar Quick sort algoritmen op het internet en maak hiervan een c# versie (je mag uiteraard zelf ook een versie ontwikkelen op basis van de info uit de presentatie)
        i. Je zal verschillende mogelijkheden vinden wat betreft “pivot selectie”
        ii. Je zal verschillende mogelijkheden vinden wat betreft “partionering”
    b. Doe ook hiervan een tijdsperformantie meting met verschillende input.

    c. Maak ook hier een versie om de Auto’s te sorteren, echter nu een verbetering zodat de logica die bepaalt **wat er moet worden vergeleken** (km stand, bouwjaar,Model…) **niet meer in het sorteeralgoritme** zit **maar wel in de Auto klasse** zelf. (tip je zal moeten gebruik maken van operator overloading)

7. Merge Sort
    a. Bouw het merge sort algoritme aan de hand van recursie.