Laaturaportti

Projekti, Tiimi 13

Antti Keronen
Juuso Käyhkö
Severi Arpinen

Sisällysluettelo

1. Dokumentti ja tekoälyn käyttö

Käyttäjä	Aika	Lisäys
AK	23.04.2025	Raportin luonti
AK	23.04.2025	Tehtävät ja moduulit
AK	23.04.2025	Testaustrategia ja työkalut
AK	23.04.2025	Codegrade toimivat testit läpi
AK	27.04.2025	Viimeisten testien suoritus ja uusien lisäys, testirivien lisäys ja dokumentin viimeistely
AK	27.04.2025	Todistusaineistot testien läpäisystä / läpäisemättömyydestä

Tekoälyn käyttö: Tekoälyä Chatgpt käytetty laskemaan koodirivit github tiedostoista.

2. Johdanto

Testaus vastaavana projektin viimeistelyssä toimii Antti Keronen. Tehtävänä luoda testejä ja varmistaa, että ohjelma toteuttaa vaaditut toiminnallisuudet. Tehtävänä myös dokumentoida testaus.

3. Testauksen tiivistelmä

Testaus aloitettiin “Smoke testingillä” eli sattumanvaraisesti käymällä läpi eri ominaisuuksia eri syötteillä varmistaen, että koodi on valmis oikeaa testausta varten. Kaikki ohjelman toiminnot läpäisivät testauksen onnistuneesti 27.4. Linkitetyn listan osalta tiedot luettiin ja tallennettiin oikein sekä normaalisti että käänteisessä järjestyksessä. Lista pystyttiin lajitelemaan sekä nousevasti että laskevasti, myös tilanteissa, joissa lista oli tyhjä tai sisälsi vain yhden alkion. Binääripuun testauksessa varmistettiin, että puuhun voi lisätä ja poistaa alkioita, ja että puu osaa tasapainottaa itsensä automaattisesti. Leveys- ja syvyyshaku toimivat oikein, ja haku tietyllä arvolla onnistui. Graafissa kaikki toiminnot, kuten tiedon lukeminen, noden ja kaaren lisääminen sekä noden poistaminen, toimivat moitteettomasti. Lisäksi ohjelma läpäisi kaikki testit yhtäaikaisesti, ja manuaalinen testaus eri syötteillä ei paljastanut virheitä. Myös CodeGrade arviointi meni läpi jokaisen viikon osalta, paitsi L11, jossa itse testien tasapainotuksessa oli käytetty eri metodeita.

4. Käytetty testausstrategia

Projektin testaus toteutettiin yhdistämällä automaattinen ja manuaalinen testaus. Testaamiseen käytettiin pääasiassa VsCodea, jolla luotiin yksikkötestejä ja ajettiin testit, jotta nähtiin niiden toimivuus. CodeGrade työkalu mahdollisti testien automaattisen testauksen asiakkaan luomilla testeillä. Testausstrategia sisälsi sekä yksikkötestauksen että integraatiotestauksen. Yksikkötestauksessa tarkasteltiin erillisten toimintojen, kuten alkion

lisäyksen, poistamisen, lajittelun ja haun toimivuutta linkitetyssä listassa, binääripuussa ja graafirakenteessa. Integraatiotestauksessa varmistettiin, että eri komponentit toimivat oikein yhdessä. Esimerkiksi tiedoston lukeminen, tietojen tallentaminen ja lajittelu testattiin kokonaisuutena.

5. Testausympäristö ja käytetyt työkalut

Testaus ympäristönä toimi Vscode ja Codegrade. Loimme erillisen tiedoston testeille, joka ajetaan makefilen avulla. Käytettiin C-kielen assert kirjastoa, jolla testit voitiin suorittaa. Lopuksi kun ohjelma läpäisi testit, voitiin varmistaa, että se läpäisee myös viikottaiset codegraden testit. Testauksessa käytetty *valgrind* työkalua. Ohjelma on aloitettu *valgrind ./projekti* komennolla. Alla oleva kuvankaappaus on ohjelman lopettamisen jälkeen tulostuva yhteenveto ympäristöstä ja käytetystä muistista.

Testaus työkalujen tiedot:

ASUS ExpertBook Windows 11

Visual Studio Code

Version: 1.99.0 (user setup)

Commit: 4437686ffebaf200fa4a6e6e67f735f3edf24ada

Date: 2025-04-02T21:35:19.530Z

Electron: 34.3.2

ElectronBuildId: 11161073

Chromium: 132.0.6834.210

Node.js: 20.18.3

V8: 13.2.152.41-electron.0

OS: Windows_NT x64 10.0.26100

Yksikkötesteissä käytetty ASSERT.H kirjastoa.

Muistivuodot testattu valgrind työkalulla VSCodessa käskyllä valgrind ./projekti. Muistivuotoja ei löytynyt.

```
==16674== HEAP SUMMARY:
==16674==     in use at exit: 0 bytes in 0 blocks
==16674==   total heap usage: 73 allocs, 73 frees, 15,816 bytes allocated
==16674==
==16674== All heap blocks were freed -- no leaks are possible
==16674==
```

6. Testitapaukset

Tapaus	Kuvaus	Testimuoto	Aika	Tulos
Alkion lisäys	Lisätään alkio (node). Katsotaan että lisäys	Yksikkötesti	27.04	Läpi

	onnistuu, eikä node poistu.			
Tiedon luku linkitettyyn listaan	Ohjelma lukee tiedot tiedostosta linkitettyyn listaan	Yksikkötesti	27.04	Läpi
Tiedon tallennus linkitettyyn listaan	Ohjelma tallentaa luetut tiedot linkitettyyn listaan	Yksikkötesti	27.04	Läpi
Tiedon tallennus takaperin linkitettyyn listaan	Ohjelma tallentaa luetut tiedot takaperin linkitettyyn listaan	Yksikkötesti	27.04	Läpi
Linkitetyn listan sorttaus laskevasti	Ohjelma sorttaa linkitetyn listan tiedot laskevasti, ensiksi määrän, sitten aakkosten perusteella	Yksikkötesti	27.04	Läpi
Linkitetyn listan sorttaus laskevasti tyhjällä listalla	Ohjelma sorttaa linkitetyn listan tiedot tyhjällä listalla laskevasti	Yksikkötesti	27.04	Läpi
Listan sorttaus laskevasti yhdellä alkiolla	Ohjlema sorttaa linkitetyn listan tiedot laskevasti yhdellä alkiolla	Yksikkötesti	27.04	Läpi
Listan sorttaus nousevasti	Ohjelma sorttaa linkitetyn listan tiedot nousevasti ensiksi määrän, sitten aakkosjärjestyksen perusteella	Yksikkötesti	27.04	Läpi
Listan sorttaus nousevasti tyhjällä listalla	Ohjlema sorttaa linkitetyn listan tiedot tyhjällä listalla	Yksikkötesti	27.04	Läpi
Listan sorttaus nousevasti yhdellä alkiolla	Ohjelma sorttaa linkitetyn listan tiedot nousevasti yhdellä alkiolla	Yksikkötesti	27.04	Läpi

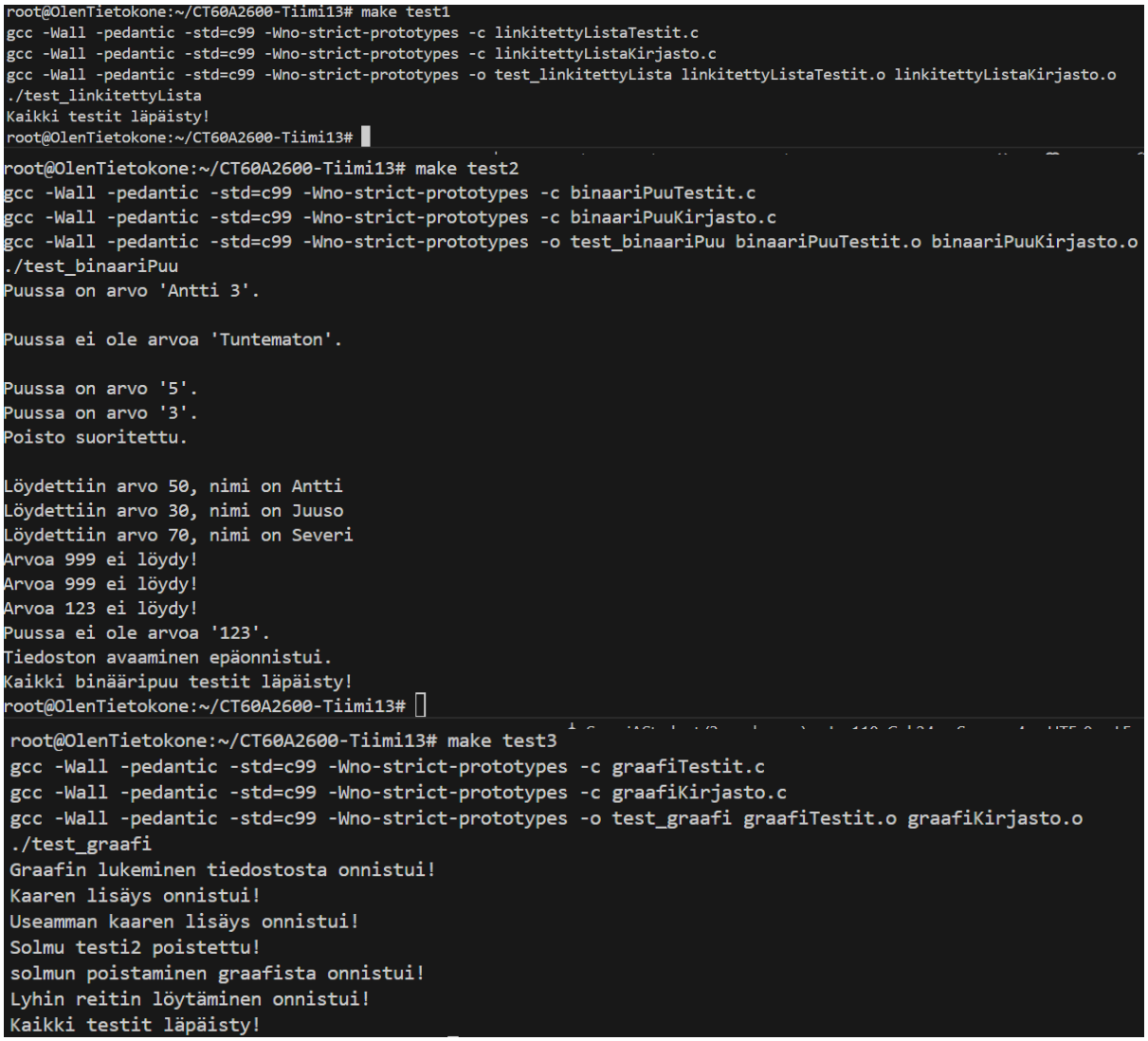
Binääripuun alkioiden järjestys	Ohjelma tasapainottaa binääripuun alkiot oikeaan järjestykseen	Yksikkötesti	27.04	Läpi
Binääripuun alkioiden lisäys	Ohjelma lisää alkion binääripuuhun käyttäjän käskystä	Yksikkötesti	27.04	Läpi
Binääripuun alkioiden olemassa olo	Ohjelma tarkistaa onko binääripuussa alkioita.	Yksikkötesti	27.04	Läpi
Binääripuun leveyshaun jonon toiminta	Ohjelman jonotoiminto toimii leveyshakua varten.	Yksikkötesti	27.04	Läpi
Binääripuun Leveyshaun toiminta	Ohjelman binääripuun leveyshaku toimii käyttäjän antamilla tiedoilla	Yksikkötesti	27.04	Läpi
Binääripuun Syvyyshaun toiminta	Ohjelman binääripuun syyvyshaku toimii käyttäjän antamilla tiedoilla.	Yksikkötesti	27.04	Läpi
Binääripuun noden Poiston toiminta	Ohjelmalla voi poistaa käyttäjän haluaman noden.	Yksikkötesti	27.04	Läpi
Binaarihaun toiminta	Ohjelmalla voi etsiä tiettyä nodea käyttäjän antamilla tiedoilla, joko määrällä tai nimellä	Yksikkötesti	27.04	Läpi
Binääripuun Tasapainotus toiminta	Ohjleman binääripuutoiminto tasapainottaa itse itsensä datan lisäyksen tai	Yksikkötesti	27.04	Läpi

	poiston yhteydessä			
Graafin tietojen lukeminen tiedostosta	Ohjelma lukee tiedot tiedostosta graafiin.	Yksikkötesti	27.04	Läpi
Kaaren lisäys graafiin	Ohjelma lisää graafin nodejen välille kaaren käyttäjän antamilla tiedoilla	Yksikkötesti	27.04	Läpi
Noden lisäys graafiin ja graafin päivitys	Ohjelma lisää graafiin noden ja päivittää graafin käyttäjän antamilla tiedoilla	Yksikkötesti	27.04	Läpi
Graafin noden poistaminen	Ohjelma poistaa käyttäjän haluaman noden graafista	Yksikkötesti	27.04	Läpi
Graafin nodejen välisen lyhimmän reitin etsiminen ja sen tallentaminen tiedostoon	Ohjelmaan voi syöttää nodejen nimiä ja graafi etsii näiden välisen lyhimmän reitin ja tallentaa tiedostoon	Yksikkötesti	27.04	Läpi
Kaikki testit läpi yhtä aikaa	Ohjelman kaikki testit läpäistään samanaikaisesti	Integraatiotesti	27.04	Läpi
Ohjelman manuaalinen testaus eri syötteillä	Ohjelman kaikkia toimintoja testataan manuaalisesti virheiden löytämiseksi.	Integraatiotesti	27.04	Läpi
Muistivuotojen testaus	Testataan löytyykö muistivuotoja valgrind käskyllä	Muistivuodon testaus	27.04	Läpi
Codegrade viikottain läpi	Codegrade läpäistään virheittä.	Integraatiotesti	27.04	Ei läpi (L11 virhe)

6.1. **Toiminnalliset testit**

Tapaus	**Aika**	**Tulos**
Alkion lisäys	27.04	Läpi
Tiedon luku	27.04	Läpi
Tiedon tallennus	27.04	Läpi
Tiedon tallennus takaperin	27.04	Läpi
Listan sorttaus laskevasti	27.04	Läpi
Listan sorttaus laskevasti tyhjällä listalla	27.04	Läpi
Listan sorttaus laskevasti yhdellä alkiolla	27.04	Läpi
Listan sorttaus nousevasti	27.04	Läpi
Listan sorttaus nousevasti tyhjällä listalla	27.04	Läpi
Listan sorttaus nousevasti yhdellä alkiolla	27.04	Läpi
Binääripuun alkioiden järjestys	27.04	Läpi
Binääripuun alkioiden lisäys	27.04	Läpi
Binääripuun alkioiden olemassa olo	27.04	Läpi
Leveyshaun jonon toiminta	27.04	Läpi
Leveyshaun toiminta	27.04	Läpi
Syvyyshaun toiminta	27.04	Läpi
Tasapainotuksen toiminta	27.04	Läpi
Poiston toiminta	27.04	Läpi
Binäärihaun toiminta	27.04	Läpi
Graafin tietojen lukeminen tiedostosta	27.04	Läpi
Kaaren lisäys graafiin	27.04	Läpi
Noden lisäys graafiin ja graafin päivitys	27.04	Läpi
Graafin noden poistaminen	27.04	Läpi
Graafin nodejen välisen lyhimmän reitin etsiminen ja sen tallentaminen tiedostoon	27.04	Läpi
Kaikki testit läpi yhtä aikaa	27.04	Läpi

Ohjelman manuaalinen testaus	27.04	Läpi
Codegrade läpi	27.04	Ei Läpi (L11 virhe)



6.2. Laadulliset testit (ei-toiminnalliset)

Tapaus	Aika	Tulos
Tiedoston virheen käsittely	27.04	Läpi
Muistivuodot	27.04	Läpi
Tiedoston sisältö väärä	27.04	Läpi

Suorituskyky (suuri tieto määrä)	27.04	Läpi

6.3. Muut testit

7. Riskiarvio

Päälimmäinen riski on ohjelman laajennettavuus ja sen mahdollinen vaikeus.

Tiedostojen lukeminen on tehty ainoastaan tietynlaisen datan lukemiseen (nimi;määrä), joten mikäli sillä yritetään lukea väärälaisia tiedostoja, voi siitä koitua ongelmia. Riskinä tällä hetkellä on myös, että ohjelma on vääränlainen, eikä toteuta asiakkaan vaatimuksia, sillä asiakkaan luoma testi ei mennyt koodilla läpi. Ohjelma kylläkin toimii lokaalisti. On mahdollista, että ohjelmaa voi myös olla vaikea käyttää, jos ei ymmärrä käyttöliittymässä käytettyjä termejä.

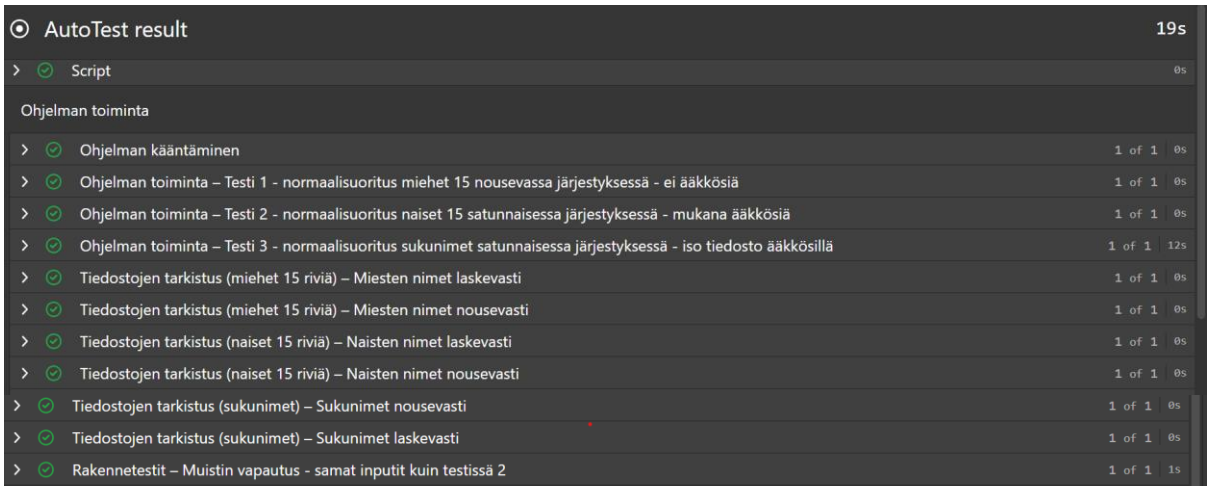
Riski	**Todennäköisyys**	**Vaikutus**	**Mahdollinen ratkaisu**
Ohjelman laajennettavuus	Keskitaso	Keskitaso	Ohjelman rakentaminen laajennettavaksi ja modulaariseksi
Ohjelma lukee ainoastaan tietynlaisia tiedostoja (nimi;määrä) ja ohjelmassa koituu ongelmia, jos tiedosto ei ole oikeassa formaatissa	Keskitaso	Keskitaso	Ohjelmaan voisi lisätä lukutapoja eri tiedostojen lukemiseen
Koodissa on virhe, mutta luullaan että CodeGradessa on virhe	Suuri	Suuri	Oma koodi on yleensä ensimmäinen kohta, jota tarkistaa, mutta L11 virhe on todennäköisesti CodeGradessa.
Ohjelman käyttöliittymän termien ymmärtämättömyys ja sen myötä vaikea käyttö	Matala	Matala	Käytetään yksinkertaisia sanoja tai lisätään selitykset varmuuden vuoksi.

8. CodeGrade-raportti

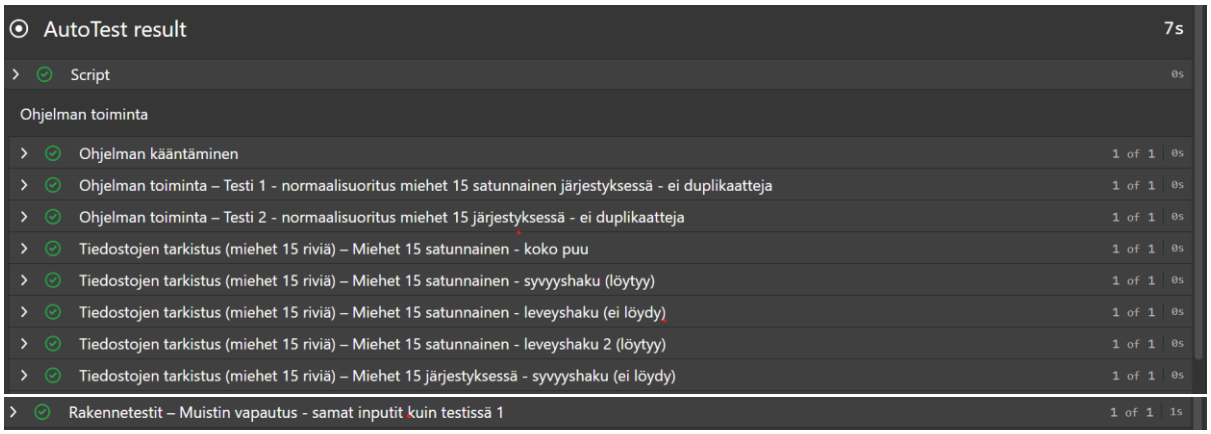
Ohjelma läpäisee kaikki asiakkaan luomat testit, lukuun ottamatta viallista testitapausta L11 kohdassa, jossa testitapauksen tasapainotus oli viallinen. Koodi siis läpäisee 100% toimivat CodeGrade testit.

L08

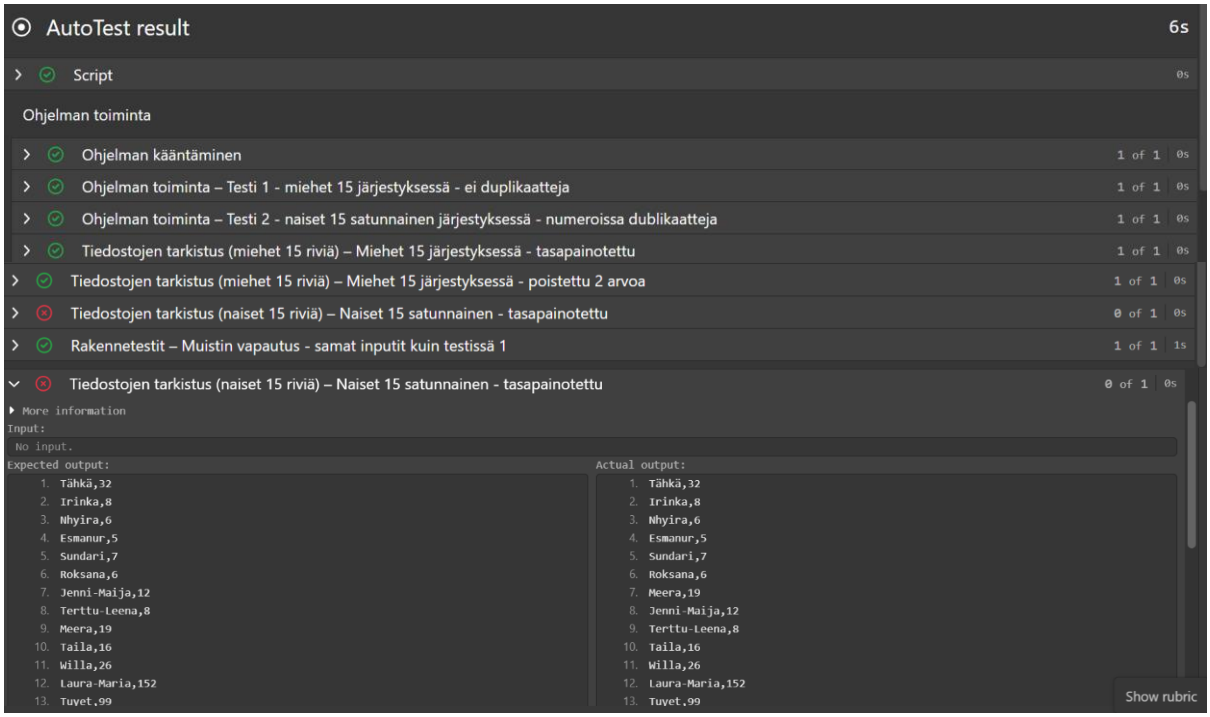
L09



L10



L11



L12

9. Testauksen aikana korjatut virheet

Lopputestauksessa havaitsimme ja korjasimme seuraavat virheet:

Ohjelma lopettaa itse itsensä, jos käyttäjä lukee graafiin tiedot, tyhjentää graafin ja yrittää tulostaa graafin. Virheeksi löytyi, että osoitin osoitti muistipaikkaan, jota ei enää ollut olemassa.

Graafin solmun poistamisessa ei tarkistettu oliko poistettava solmu oikeasti graafissa olemassa.

10. Testien Rivimäärä

Luento	Rivimäärä
L08	34
L09	151

L10	282
L11	363
L12	476
L13	543
L14	543