Hvorfor testes der? Hvilken testtype bruges, og hvorfor?

For at sikre at programmet kører stabilt, og laver de korrekte udregninger, testes programmet for forskellige testcases. Der er forskellige metoder at teste dette på, hvoraf testen i denne rapport er blackbox testing. Denne testtype behandler en mængde testcases, som beskrevet i tabellen, hvor et bestemt input har et forventet output, og programmet testes derefter. Hvis det forventede output stemmer overens med det reelle output fra programmet, kan man der ud fra konkludere på testen.  
En anden testtype kunne være CU-test, som er et indlejret test-system, hvor testcases bliver oprettet som kode i programmet, derefter vil testen bliver kørt på compile-time og resultater fra testcases bliver printet ud. Begrundelsen for brug af blackbox testing er, at der i dette program bliver printet gennem processen, og alle valg i programmet bliver foretaget af brugeren. Deraf vil en blackbox testing give et godt billede af, om testen og input fra brugeren stemmer overens.

CASE 1:

I denne første testcase, vil inputtet til valg af attraktioner være 1, 5 og 9. Ved brug af disse tal, vil et forventet output være Aalborghus\\_Slot for 1, Springeren\\_-\\_Maritimt\\_Oplevelsescenter for 5 og Nordkraft for 9. Herefter ville valget af attraktioner afsluttes, ved input 0.

Output ved førte del af testcasen blev følgende: ”Tilfoejet attraktion: Aalborghus\\_slot”, ”Tilfoejet attraktion: Springeren\\_-\\_Maritimt\\_Oplevelsescenter” og ” Tilfoejet attraktion: Nordkraft”. Herved gav den første del et korrekt output. Ved afsluttelse af valg af attraktion, tilføjes disse attraktioner til ruten, og næste step er tilføjelse af interessante, nærliggende attraktioner. Eftersom de valgte attraktioner alle ligger tæt på havnen i Aalborg, vil andre interessante attraktioner være Utzon Centeret, Havnefronten og Friis, da disse alle ligger tæt på en tiltænkt rute fra Nordkraft til Springeren, hvor Aalborghus Slot også besøges.\newline

Her foreslår programmet følgende: 1: Utzon\\_Centeret, 2: Friis\\_Aalborg\\_Citycenter og 3: Havnefronten. Dette er et korrekt output efter de attraktioner der blev valgt. Disse er alle tre er inden for en afstand af 100 meter fra enten den nuværende rute eller de valgte attraktioner. Efterfølgende skal brugeren selv vælge, om han vil tilføje disse attraktioner til ruten. I dette tilfælde bliver inputtet 1 og 2, for tilføjelse af Utzon\\_Centeret og Friis\\_Aalborg\\_Citycenter. Outputtet blev ”Tilfoejet attraktion: Utzon\\_Centeret” og ”Tilfoejet attraktion: Friis\\_Aalborg\\_Citycenter”. Dette stemmer overens med det forventede output, og tilføjelsen afsluttes med input 0. Herefter vil ruten blive dannet, og alle attraktioner valgt vil blive printet ud som ”Din rute”. Heraf vil der vises Aalborghus Slot, Springeren, Nordkraft, Utzon Centeret og Friis. Disse vil sorteres efter hvornår på ruten de besøges, hvor startpunktet vil blive printet dobbelt, som både start-attraktion og slut-attraktion. Siden startattraktionen er Aalborghus\\_Slot, skal ruten blive Aalborghus\\_Slot, Utzon\\_Centeret, Friis\\_Aalborg\\_Citycenter, Nordkraft, Springeren\\_-\\_Maritimt\\_Oplevelsescenter og Aalborghus\\_Slot. Dette er også tilfældet, da vores output er magen til det forventede:\newline

Din rute:\newline

Aalborghus\\_Slot\newline

Utzon\\_Centeret\newline

Friis\\_Aalborg\\_Citycenter\newline

Nordkraft\newline

Springeren\\_-\\_Maritimt\\_Oplevelsescenter\newline

Aalborghus\\_Slot\newline

Herefter er der også et output der beskriver rutens længde, hvor et forventet resultat er udregnet gennem movable-type.co.uk, hvilket afrundet er 5.59km. Ifølge ouputtet er længden 5.61km, hvilket er omkring 20 meter fra det forventede resultat. Dette er et fint resultat, som viser at programmet i dette tilfælde har en fejlberegning på 0.31%. Denne fejlmargin er fin, da vi har afrundet vores tal.

CASE 2:

Da vi bruger samme input i denne case, vil testen være den samme, indtil tilføjelsen af interessante attrationer skal have input. Dette input testes med et input der er højere end antallet af forslag, hvilket i dette tilfælde vil være 4. Her kommer en fejlmelding fra programmet: ”Tallet svarer ikke til en attraktion”, og der promptes efter nyt validt input. Ved indtastning af samme tal flere gange, kommer den forventede fejlmelding ”Du har allerede indtastet denne attraktion. Proev igen”.

CASE 3:

Ligeom i CASE 2, er inputtet det samme indtil tilføjelsen af interessante attraktioner, hvor der i denne testcase testes for input af bogstaver, tegn og ord. I dette tilfælde vil ”a”, ”!” og ”test” alle tre printe ”Fejlindtastning – proev igen”, og derefter promte efter nyt validt input. I en tidligere version printede programmet denne sætning for hvert tegn og bogstav inputtet bestod af. Så i ”test” blev der printet fire ”Fejlindstastning – proev igen”.

CASE 4:

Igen her blev testen udført med det samme input som i CASE 2 bortset fra, at inputtet til tilføjelsen af interessante attraktioner vil være ”0”, for afsluttelse af ruten uden tilføjelse af attraktioner. Her kører programmet videre, og giver den endelige rute:

Din rute:\newline

Aalborghus\\_Slot\newline

Nordkraft\newline

Springeren\\_-\\_Maritimt\\_Oplevelsescenter\newline

Aalborghus\\_Slot\newline

Rutens længde ville forventet afrundet være 5.54, og outputtet fra programmet siger 5.56, som igen er omkring 20 meter længere.

CASE 5:

Ved første prompt viser den antallet af attraktioner, og hvis alle attraktioner vælges, bliver der ikke promptet for en attraktion udover antallet. Dette vil sørge for, at en bruger ikke kan vælge flere attraktioner ind databasen er tilskrevet. Derfor vil programmet selv steppe videre ved valg af alle attraktioner.

CASE 6:

I denne testcase vil det første prompt testes for, hvorvidt det er muligt at indtaste en samme attraktion flere gange. Forventningen er, at en fejlmelding forekommer ved mere end én indtastning for samme attraktion. Ved indtastning af 1 to gange i træk, kom fejlmeldingen ”Du har allerede indtastet denne attraktion. Proev igen”. Derefter testes for, hvorvidt man kan skrive to forskellige tal, hvoraf det første blive skrevet to gange, med det andet tal i mellem. Dvs. 1, 2 og 1 igen, her forventes samme fejlmelding ved anden indtastning af 1. Programmet printede derefter samme fejlmelding.

CASE 7:

I testcase 7 vil det første prompt igen blive testet, denne gang for input af tegn, bogstaver og ord. Igen testes med ”a”, ”!” og ”test”. Her forventes samme resultat som i testcase 3, hvor programmet i dette tilfælde vil printe fejlmeldingen ”Fejlindstastning – proev igen”. Her var resultatet som forventet, i alle tre tilfælde blev der printet ”Fejlindstastning – proev igen”. Ved indtastning af et tal højere end antallet af attraktioner, forventes fejlmeldingen ”Tallet svarer ikke til en attraktion”. Dette er også tilfældet, da programmet giver den rigtige fejlmelding.

CASE 8:

I denne case testes programmet for intet valg af attraktioner, ved at første input er 0. Forventningen i denne test er, at programmet blot afsluttes. Dette er også tilfældet, dog havde en rettelse været nødvendig, da det originale program blot udskrev en rute uden attraktioner, med en afstand på 100000.

Efter disse testcases kan det påvises, at programmet kører som planlagt, efter et par rettelser. Der er testet for, hvorvidt programmet giver en rute ved korrekt input, muligheder for forkerter input, hvorvidt programmet giver en korrekt rutelængde, samt om attraktionerne tilføjet som interessante attraktioner er korrekte. Brugeren skulle ikke have mulighed for at give forkert input, rutelængden er med minimal fejlagtighed, korrekte interessante attraktioner bliver tilføjet, og ved et korrekt input vil en rute altid blive oprettet.