1. One thing I struggled with was coming up with the code that would return error when a y or n was not given as an input. I kept trying to write it as a or logical statement. It wasn’t until I wrote it down on my iPad and divided up the clauses that I realized it should be an and not an or.
2. Test data that will run on my program
   1. Input a negative age such as –12, this will return an error at the end. The reason is because the code should not accept negative values for age.
   2. Inputting anything that’s NOT a y or n will return an error message e.g. (x). The reason is because the code shouldn’t accept answers that are NOT y or n.
   3. Inputting an empty spring for Destination will return an error message e.g. (“”). The reason is because the code must have a string with characters in it.
   4. Inputting a negative number of boundaries will return an error message e.g. -2.. The reason is because the code shouldn't accept a negative number for boundaries.
   5. 64-year-old non student with 1 boundary should return $2.00. This should be ran because this is the base case with no discounts.
   6. 22-year-old student with 3 boundaries should return $3.10. This should be ran because this is the base case with no discounts.
   7. 17-year-old non student with 1 boundary should return $0.65. This should be ran because it tests if the minor discount applies.
   8. 23-year-old student with 0 boundaries crossed should return $0.65. This should be ran because it tests if the student discount works.
   9. 65-year-old senior non student with 0 boundaries crossed will return $0.45. This should be ran because it tests if the senior discount is applied.
   10. 72-year-old senior student with 1 boundary crossed will return $0.65. This should be ran because it would tests if the student discount applies to seniors.
   11. 83-year-old senior non student with 3 boundaries crossed should return $1.60. This was ran to test the senior discount at a higher number of boundaries.