```
Your answers here should be to the coding part of the assignment only.

How much time did you spend on the assignment?
4 hours

Who did you work with and how?
I worked with Richa to clarify the first few questions.

What resources did you use, and which ones were helpful/unhelpful?
N/A

Give a few sentences reflecting on your experience with the assignment,
such as whether you found it useful, boring, helpful, confusing, and why.
Suggestions for improving it are always welcome.
I found this assignment to be very confusing, but it helped to clarify the uses of while and "→".
I think clearer instructions would help.
```

```kotlin
import kotlin.test.*
// Ad Personalization
//
// Congratulations! You've been invited to do the following
// online assessment for a co-op with tech giant Oodle.
//
// 1. Add constants (such as Female, Male, and Nonbinary) to
//    the Gender enum. It is your choice what to add.
// 2. After reading the below code, decide which ads should be
//    returned for different demographics. Make sure that all
//    ads will be returned for some combination of inputs. Add at
//    least two more age-based constants. Do not use randomness.
//    Choose the highest-revenue ad you think someone in that
//    demographic would click on.
// 3. Add properties minAge and maxAge to Ad, and set values
//    for each ad. For example, minAge for the dating ad should
//    be MIN_ADULT_AGE.
// 4. Create a function runTests() that consists of tests of fetchAd
   ().
//    Use kotlin.test.assertEquals() for each test. There should
//    be at least one test for each of the provided ads.
// 5. Implement fetchAd(). You should use both "if" and "when".
//    Run your tests, revising your code if they fail valid tests
//    and your tests if you find any mistakes in them.
// 6. Create a new data class named "Person". A person should have an
//    age (Int), gender (Gender), and income (Int). Use your judgment
//    as to which should be changeable.
// 7. Write a new fetchAd() method (without removing the original one)
//    that takes a single parameter of type Person and returns an Ad.
//    Instead of duplicating the code in your original fetchAd()
   method,
//    have your new method call your old method, passing the
   appropriate
//    properties as arguments.
// 8. Add 3-5 tests of the new fetchAd() method to runTests().
// 9. State in a comment whether you satisfied all of the above
//    requirements and all of your tests passed.

enum class Gender {
    Female, Male, Nonbinary
    // 1. Add Gender constants here, such as Female, Male, and
   Nonbinary.
}

// Age-based constants
const val MIN_AGE_FOR_PERSONALIZATION = 13
const val MIN_ADULT_AGE = 18
const val MIN_MID_ADULT_AGE = 45
const val MIN_SENIOR_AGE = 67
// 2. Add at least two more age-based constants.

fun runTests() {
    assertEquals(Ad.Diet, fetchAd(Gender.Female, 65, 0))
```

```kotlin
    assertEquals(Ad.Car, fetchAd(Gender.Male, 18, 1000000))
    assertEquals(Ad.Beauty, fetchAd(Gender.Nonbinary, 66, 200000))
    assertEquals(Ad.Dating, fetchAd(Gender.Female, 18, 0))
    assertEquals(Ad.Pet, fetchAd(Gender.Nonbinary, 100, 1500000))
    assertEquals(Ad.PetToy, fetchAd(Gender.Male, 13, 10000))
    assertEquals(Ad.Lego, fetchAd(Gender.Female, 13, 0))
    assertEquals(Ad.Pokemon, fetchAd(Gender.Nonbinary, 13, 120000))
    assertEquals(Ad.Retirement, fetchAd(Gender.Female, 100, 80000))
    assertEquals(Ad.Work, fetchAd(Gender.Male, 18, 0))
    fetchAd(Gender.Female, 13, 0)
    fetchAd(Gender.Male, 45, 30000)
    fetchAd(Gender.Nonbinary, 67, 20000)
    fetchAd(Gender.Female, 8, 0)
    fetchAd(Gender.Male, 102, 10000)
    println("All Tests Passed.")
}

fun main(){
    runTests()
}
/**
 * Ads that may be shown to users.
 *
 * @property text the text of the ad
 * @property revenue the number of cents earned per click
 */
// 3. Add properties minAge and maxAge.
enum class Ad(val text: String, val revenue: Int, val minAge: Int,
   val maxAge: Int) {
    Diet("Lose weight now!", 5, 18, 66),
    Car("Buy a new car!", 5, 18, 66),
    Beauty("Improve your looks!", 5, 18, 66),
    Dating("Meet other singles!", 4, 18, 100),
    Pet("Get a pet!", 3, 13, 100),
    PetToy("Buy your pet a toy!", 2, 13, 100),
    Lego("Get more bricks!", 2, 13, 44),
    Pokemon("Gotta catch 'em all!", 2, 13, 44),
    Retirement("Join AARP", 2, 67, 100),
    Work("Apply for a job at Oodle!", 2, 18, 66)
}

data class Person(val gender: Gender, val age: Int, val income: Int
   ){}

/**
 * Fetches an ad based on the user's [gender], [age], and
 * [income], unless the age is under [MIN_AGE_FOR_PERSONALIZATION],
 * in which case no personalization is permitted.
 */
fun fetchAd(gender: Gender, age: Int, income: Int): Ad {
    if (age < MIN_AGE_FOR_PERSONALIZATION) {return Ad.Lego}
    if (age ≥ MIN_AGE_FOR_PERSONALIZATION && age < MIN_ADULT_AGE) {
        return when (gender) {
```

```
101             Gender.Female → {Ad.Lego}
102             Gender.Male → {Ad.PetToy}
103             Gender.Nonbinary → {Ad.Pokemon}
104         }
105     }
106     if (age ⩾ MIN_ADULT_AGE && age < MIN_MID_ADULT_AGE) {
107         return when(gender) {
108             Gender.Female → {
109                 if (income ⩾ 30000) {Ad.Car}
110                 else if (income ⩾ 15000) { Ad.Pet}
111                 else {Ad.Dating}
112             }
113             Gender.Male → {
114                 if (income ⩾ 30000) {Ad.Car}
115                 else if (income ⩾ 15000) { Ad.Pet}
116                 else {Ad.Work}
117             }
118             Gender.Nonbinary → {
119                 Ad.Diet
120                 Ad.Beauty
121                 Ad.Dating
122                 Ad.Lego
123                 Ad.Pokemon
124                 Ad.Work
125                 if (income ⩾ 30000) {Ad.Car}
126                 else if (income ⩾ 15000) { Ad.Pet}
127                 else {Ad.PetToy}
128             }
129         }
130     }
131     if (age ⩾ MIN_MID_ADULT_AGE && age < MIN_SENIOR_AGE) {
132         return when(gender) {
133             Gender.Female → {
134                 if (income ⩾ 50000) {Ad.Car}
135                 else if (income ⩾ 20000) {Ad.Pet}
136                 else { Ad.Diet}
137             }
138             Gender.Male → {
139                 Ad.Diet
140                 Ad.Beauty
141                 Ad.Dating
142                 if (income ⩾ 50000) {Ad.Car}
143                 else if (income ⩾ 20000) {Ad.Pet}
144                 else {Ad.PetToy}
145             }
146             Gender.Nonbinary → {
147                 if (income ⩾ 50000) { Ad.Beauty}
148                 else if (income ⩾ 20000) {Ad.Pet}
149                 else {Ad.PetToy}
150             }
151         }
152     }
153     if (age ⩾ MIN_SENIOR_AGE && age ⩽ 100) {
```

```
154         return when(gender) {
155             Gender.Female → {
156                 Ad.Dating
157                 if (income ⩽ 23000) {
158                     Ad.Pet
159                     Ad.PetToy
160                 }
161                 else {Ad.Retirement}
162             }
163             Gender.Male → {
164                 Ad.Dating
165                 if (income ⩾ 23000) {
166                     Ad.Pet
167                     Ad.PetToy
168                 }
169                 Ad.Retirement
170             }
171             Gender.Nonbinary → {
172                 if (income ⩾ 23000) {
173                     Ad.Pet
174                 }
175                 else {Ad.Retirement}
176             }
177         }
178     }
179     else {return Ad.Car}
180 }
181     // 5. Serve each of the above for at least one combination
182     // of inputs. If age is less than MIN_AGE_FOR_PERSONALIZATION,
183     // you must not use gender or income.
184 fun fetchAd(person: Person){
185     fetchAd(person.gender, person.age, person.income)
186 }
187
188 //I satisfied all of the above requirements and all of my tests
    passed
```