The CS Majors

Adam Davis (gte459u)
Namit Bhatia (gte276u)
Nimish Gautam (gte671u)

CS Project: Milestone 6
The CS Majors – Test Plan for M6

Adam Davis  gte459u
Nimish Gautam  gte671u
Namit Bhatia  gte276u

Documentation of a test plan based on the Equivalence Partitioning approach.

In equivalence partitioning, the possible cases of input are separated into different categories which are then represented by specific test cases.

To test the newly added Family class, we must think of all the possible categories of input that might show that there are errors in the code. These will be the equivalence partitions of our test plan. The family class is filled in dynamically as the user defines relationship of one person with another. For example, if the command:

\[ \text{personA hasFather: personB} \]

is issued, there are several checks that occur in the Family class and then personA is added to the appropriate family as required.

The following checks occur before adding personA (above) to a family:

1.) Does the person already have a mother defined?
   a. \textit{If yes}: Add personB above to the family consisting of personA, his/her mother, and personB.
   b. \textit{If no}: Define a new family with personB as the father and personA as the child.

2.) A very similar scenario occurs when assigning a mother to personA, where the existence of a family with personA’s father is checked.

3.) If personB is being defined as the spouse of personA, then check if there is already a spouse of personA and personB. If yes, do nothing, if no, create it.

These checks can have the following different types of errors which we need to check for.

A) There already exists a family with personA’s mother defined, in spite of which a new family is defined consisting of personA and personB (as father) when defining a father for personA using the above command.

B) If there was a family with personB defined as a father already, but the wife of personB may not necessarily be the mother of personA, personA may be erroneously added to the family to which he/she does not belong.

C) Similarly, for the definition of a spouse, we need to ensure that there doesn’t already exist a family that is uniquely identified by both the spouses that are now being defined.

D) Another possible error in the definition of a spouse can be when any one of the spouses that are currently being defined are already in a marriage but NOT in a marriage with each other, but a new family is not created.

Every case mentioned above will be considered an equivalence partition and a test case will be made for it which represents that partition. For instance, point A contains three different partitions, personA should already have a mother defined, personA should already have personB as the father defined, and thirdly, personA may already have
personB as the father as well as a mother defined. For each of these partitions we will make a case where a Person class is given information that breaks this rule and then run “PrintIt” on <Person> check as well as a “PrintIt” on Population Families to see if the code accounts for that rule and prints the proper response (that is, there should be no extra families, and the child should be assigned to the correct family). If it doesn’t, we will consider that test failed.

To test the Population class for the generateGEDCom function and readGEDCom function, we must think of all the different combinations of families that can exist, and ensure that the file that is generated outputs all the familial information, birth and death information, names, and other information into a GEDCom file that can be read by GED2Html. For testing the readGEDCom, this file is again imported back into the genealogy system and all the families and person-info is checked to ensure that there is nothing missing, nor is there any extra information. The following are the different cases of families that can exist, which can possibly be erroneous:

a) A simple nuclear family with two parents and children. – All the children should have both mother and father correctly defined, and they should all be the members of the same family, say family X.

b) Families where the father is the same and the mother is different. The generateGEDCom function should assign the correct mother to every child of these families, and not mistake one mother for the other since the fathers are the same.

c) Two separate families where the mother has children from two different fathers.

d) Another possible error is where a father is assigned to children who don’t already have a father defined but have mothers defined.

Every one of the cases mentioned above will be considered an equivalence partition and a test case will be made for it which represents that partition. Separate families will be created with one and multiple children. Also, different families will be created where the father is common but mothers different. Also, a “PrintIt” will be done on Population Families to ensure that there are no duplicate families and there are no families that are missing that should actually be present.