Equivalence Partition for FoodForMood

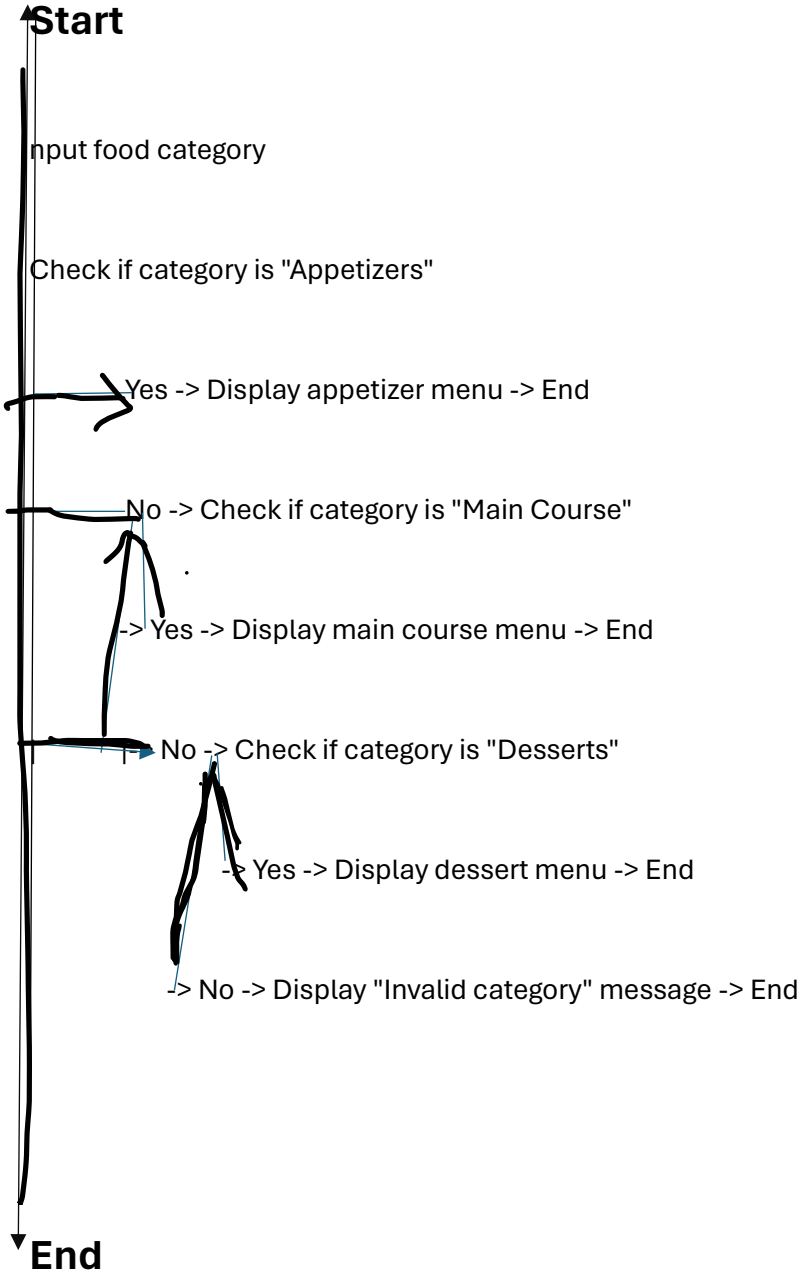
The scenario involves the equivalence partitioning of the food menu and food category for our website, FoodForMood. The website allows users to enter the name of a food item, its category, and a list of prices for different sizes or portions. According to the specifications, the food item name should be 2 to 20 characters long and can only contain alphanumeric characters. Each price can be a value between 1 and 100, and a maximum of five prices can be entered for each item. The item name should be entered first, followed by a comma, then the category, and another comma. Finally, the list of prices should be entered. Spaces should be ignored anywhere in the input.

Derived Equivalence Classes:

1. Food item name is alphanumeric (valid)
2. Food item name is not alphanumeric (invalid)
3. Food item name is less than 2 characters in length (invalid)
4. Food item name is 2 to 20 characters in length (valid)
5. Food item name is greater than 20 characters in length (invalid)
6. Category is alphanumeric (valid)
7. Category is not alphanumeric (invalid)
8. Category is less than 2 characters in length (invalid)
9. Category is 2 to 20 characters in length (valid)
10. Category is greater than 20 characters in length (invalid)
11. Price value is less than 1 (invalid)
12. Price value is in the range 1 to 100 (valid)
13. Price value is greater than 100 (invalid)
14. Price value is a whole number (valid)
15. Price value is a decimal (invalid)
16. Price value is numeric (valid)
17. Price value includes non-numeric characters (invalid)
18. Prices entered in ascending order (valid)
19. Prices entered in non-ascending order (invalid)
20. No prices entered (invalid)
21. One to five prices entered (valid)
22. More than five prices entered (invalid)
23. Food item name is first (valid)
24. Food item name is not first (invalid)
25. Category is after the food item name and before prices (valid)
26. Category is not after the food item name and before prices (invalid)
27. A single comma separates each entry in the list (valid)
28. A comma does not separate two or more entries in the list (invalid)
29. The entry contains no blanks (valid)
30. The entry contains blanks (valid)

	Test Data	Expected Outcomes	Classes Covered
1	Pizza, Noodles,10,15,20	T	1,4,6,9,11,12,14,18,21,23,25,27,29
2	Fast Food,5,7,8,15	T	1,4,6,9,11,12,14,18,21,23,25,27,29
3	Dessert,1.5,3,4	F	10,15
4	Salad,0	F	11
5	Chinse 50,60,70,80,90,100	F	13,21
6	Noodles10,12,14,15,10, 20	F	19
7	,Hala Food,5,10,15,20	F	3,25
8	Pasta,Pasta8,7,6	F	19,23
9	, , ,	F	3,7,20,25
10	Irish Food,b,2,3,4	F	4,23,25
11	Desi Food,3,5,6,7	F	2,26,28
12	High Protein,5,7,8,15,20,25	F	22
13	Pizza,Biryani, 10,15,20	F	7,10
14	Chicken Rolls,5,Fast Food	F	8,10

Basis Path



We count 8 edges and 7 nodes, so V(G) = 8 - 7 + 2 = 3.

Step 3: Determine the basis set of independent paths.

Path 1: Start - Check Appetizers - Display appetizer menu - End
Path 2: Start - Check Main Course - Display main course menu - End
Path 3: Start - Check Desserts - Display dessert menu - End
Path 4: Start - Check Appetizers - Invalid category message - End
Path 5: Start - Check Main Course - Invalid category message - End

Path 6: Start - Check Desserts - Invalid category message - End

Step 4: Prepare test cases that force execution of each path in the basis set.

Path	Input Category	Expected Result
1	Appetizers	Display appetizer menu
2	Main Course	Display main course menu
3	Desserts	Display dessert menu
4	Salads	Invalid category message
5	Drinks	Invalid category message
6	Sides	Invalid category message

With these test cases, we ensure that each path in the basis set is executed, covering all possible scenarios of input categories.