**Homework T2 Submission Document**

You can work individually or in groups of 2.

|  |  |  |  |
| --- | --- | --- | --- |
| Names: | Levi Sutton |  | Harlan Harris |

**Test Derivation for *addEmp(e:Employee)***

We will use the category-partition method to derive a set of test cases for this method.

1. **Identify what criteria must hold true for each test case to pass**

The side-effects of this method are:

The employee should be added to the list if there is an open position

1. **Identify inputs/parameters for each feature under test.**

Integer value: i

Number of employees in the list

1. **Identify the categories/characteristics for each input/parameter.**

value = some value

numOfEmployees: size

1. **Partition categories into choices**

Number of employees: 0, at least 1, partially filled, completely filled

i: < 0, 0, 1, 19, > 19, one less than the number of employees, and the number of employees

1. **Number of tests before constraints added**

28

1. **TSL Input**

The *TSL* input file is shown below.

Justification for the constraints:

|  |  |
| --- | --- |
| **Constraint** | **Justification** |
| < 0 | Cannot have less than 0 employees on staff |
| 0 | There are no employees on staff |
| 0 < x < 19 | Theres atleast one or numEmployees – 1 amount of employeees |
| 19 | All 20 employees are added |
| > 19 | More than 20 members added, still should give you 20 as the total amount |

1. **TSL Output – Test Frames**

The \_\_ test frames are show below:

numEmployees:

empty   [List empty]

0 < x < 19   [Added one or numEmployees – 1]

20   [Added all employees]

>20  [Added more than 20 employees]

1. **Test Cases**

The test cases we derived are shown below.

[Remove unused columns]

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test** | **[Put name of characteristic 1 here]** | **[Put name of characteristic 2 here]** | ***Etc.*** |  |  |  |  |  |
| 1 | Empty | Empty |  |  |  |  |  |  |
| 2 | add[0] (name) | Hourly pay |  |  |  |  |  |  |
| 3 | add[0-19] (names) | Hourly Pay |  |  |  |  |  |  |
| 4 | add[0-26] (names) | Hourly Pay |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

**Test Derivation for *removeEmployee(pos:int):Employee***

We will use the category-partition method to derive a set of test cases for this method.

1. **Identify what criteria must hold true for each test case to pass**

The side-effects of this method are:

The employee at the specified location must be removed

1. **Identify inputs/parameters for each feature under test.**

Integer value: i

Number of employees in the list

1. **Identify the categories/characteristics for each input/parameter.**

value = some value

numOfEmployees: size

1. **Partition categories into choices**

Number of employees: 0, at least 1, partially filled, completely filled

i: < 0, 0, 1, 19, > 19, one less than the number of employees, and the number of employees

1. **Number of tests before constraints added**

28

1. **TSL Input**

The *TSL* input file is shown below.

Justification for the constraints:

|  |  |
| --- | --- |
| **Constraint** | **Justification** |
| < 0 | Can’t access an index < 0 |
| 0 | Removes atleast 1 employee |
| 0 <= x <= 19 | Remove employee at any specified location |
| > 19 | Can’t acces an index > 19 |
|  |  |

1. **TSL Output – Test Frames**

The \_\_ test frames are show below:

numEmployees:

empty [List empty, returns null]

0 <= x <= 19 [Removing employee at any specified index]

1. **Test Cases**

The test cases we derived are shown below.

[Remove unused columns]

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test** | **[Put name of characteristic 1 here]** | **[Put name of characteristic 2 here]** | ***Etc.*** |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |