## Design Document for the Beetle Game; Revision and Reflection

### Team Name: Jiggly Wigglies

## Your names and roles:

1. **Facilitator: Conner Bondurant**
2. **Recorder: Tradd Schmidt**
3. **Spokesperson: Basant Phuyal**
4. **Quality Control Officer: Rusty Dotson**
5. **Process Analyst: Rusty Dotson**

Note that if there are only 4 members in your team, then the Quality Control Officer should also serve as Process Analyst.

## Basic program concept:

Place a short but precise description of what your program will do here. **Update it if necessary**.

Our program will allow up to 4 users to play the game Beetles.

## Functional Abstraction Refinement

You should not be surprised if you find that the initial specification design needs improvement. Often times, people assume how the various pieces of the program works without writing them down, which can create problems when multiple people implement the pieces and you try to integrate them later.

There are two major potential areas of revision of the functional abstraction process. The first is on the overall structure of the program, and the second is on the function specifications.

#### Overall Structure

In the space below, describe any needed changes that you need in the overall structure of the program that became evident when you tried to integrate the functions together.

|  |
| --- |
| There were certain functions that needed to return a specific type of value and they didn’t. There was different names used for the individual parts. |

#### Function Specifications

Another problem you may need to fix is the set of function specifications so that program can be integrated correctly. For example, the comments that follow the search() function explains what was wrong with that function:

|  |
| --- |
| def search( target ):  """ locate target in the list  pre: target is a valid input to the function  post: returns true if the target is in the list and false otherwise """  # We forgot to give it the list as a parameter… maybe assumed it was a global  # variable? Also, it turns out that we need the index of where the target appears  # instead of just a true of false as to whether it is in there or not |

For each function in your program, write your team's original specification first, and then any explanations on what assumptions your team made and what changes were needed so that they can be integrated correctly. Included is space for 4 functions, but you can copy the tables as many times as necessary if you have more:

Function 1:

|  |
| --- |
| def player\_count()  “““  Will ask user how many people will be playing and returns this number.  Pre: Nothing  Post: An integer between 1 and 4  ”””  # We forgot to add that it needed to return that integer |

Function 2:

|  |
| --- |
| def get\_body\_part()  “““  Will generate a random integer between 1 and 6.  Pre: None  Post: All lowercase string  ”””  # We forgot to add what is returned. |

Function 3:

|  |
| --- |
| def add\_body\_part(player, body\_part)  “““  Will check a player’s list of body parts already obtained, and if the part that was input can be added to the dictionary, it is added.  Pre: player is a dictionary for whoever’s turn it is, and this dictionary contains all the keys already set necessary for all the body parts. body\_part is an all lowercase string which is the body part that is being added to a specific player’s bug.  Post: If the body part was able to be added, the dictionary key corresponding to that body part will be changed. Otherwise the dictionary is not altered.  ”””  # As far as we know this is good |

Function 4:

|  |
| --- |
| def check\_progress(player)  “““  Will print out the count of body parts that the current player has.  Pre: player is a dictionary for whoever’s turn it is, and this dictionary contains all the keys already set necessary for all the body parts.  Post: Prints out the current count of each body part for the current player  ”””  # We forgot to add that it should return a boolean |

After discussing with your team, fill out your **updated function** specifications below. We included space for 4 functions, but you can copy the tables as many times as necessary if you have more:

Function 1:

|  |
| --- |
| def player\_count()  “““  Will ask user how many people will be playing and returns this number.  Pre: Nothing  Post: An integer between 1 and 4  :return: an integer  ””” |

Function 2:

|  |
| --- |
| def get\_body\_part()  “““  Will generate a random integer between 1 and 6.  Pre: None  Post: All lowercase string  :return: an all lowercase string  ””” |

Function 3:

|  |
| --- |
| def add\_body\_part(player, body\_part)  “““  Will check a player’s list of body parts already obtained, and if the part that was input can be added to the dictionary, it is added.  Pre: player is a dictionary for whoever’s turn it is, and this dictionary contains all the keys already set necessary for all the body parts. body\_part is an all lowercase string which is the body part that is being added to a specific player’s bug.  Post: If the body part was able to be added, the dictionary key corresponding to that body part will be changed. Otherwise the dictionary is not altered.  :return: none  ””” |

Function 4:

|  |
| --- |
| def check\_progress(player)  “““  Will print out the count of body parts that the current player has. If a player has all the body parts it will return true, if they don’t it will return false  Pre: player is a dictionary for whoever’s turn it is, and this dictionary contains all the keys already set necessary for all the body parts.  Post: Prints out the current count of each body part for the current player  :return: a boolean  ””” |

## List of Function Implementers (you can copy this information from the previous document):

|  |  |
| --- | --- |
| Team Member Name | List of Functions to Implement |
| Basant Phuyal | Function 1 |
| Tradd Schmidt | Function 2 |
| Conner Bondurant | Function 3 |
| Rusty Dotson | Function 4 |
|  |  |

**(See next page for the design document reflection)**

## Design Document Reflection

After reaching a consensus, briefly describe at least three assumptions your team made about the program structure or function specifications that may have contributed to challenges in the implementation. You may add as many points as needed because the box will increase in size.

|  |
| --- |
| 1. We assumed that all body parts would be checked 2. We assumed that some values would be returned |

What technique did the team use to create the first specification document (i.e. drawing out the "flow" of the program on a sheet of paper and/or a Google shared document?) As a team, briefly describe at least one advantage and at least one disadvantage to that technique with regards to the design process.

|  |
| --- |
| We started from the beginning of the game and walked through each process we thought would need a function. One advantage is it ensures we go through the entire game and don’t miss a piece of the game, but this also means that some functions might be very large and not broken down that much. |

What techniques would each team member like to try in the future for the functional abstraction process and why? List out the team member's name and the technique he/she would like to try.

|  |
| --- |
| Tradd: Try to break the functions down a little more  Conner: Make sure only code functions that all functions for that specific function you are working on are finished.  Rusty: Use classes to replace some functions  Basant: Draw out the problem |

Write in the space below the team's response to the revision process. How well did it work? What aspects of the design process did it highlight as necessary?

|  |
| --- |
| Name: Tradd  Response:  Name: Rusty  Response:  Name: Conner  Response:  Name: Basant  Response: |