Exemplar for Internal Achievement Standard

Digital Technologies Level 1

This exemplar supports assessment against:

Achievement Standard 91075

Construct a plan for a basic computer program for a specified task

An annotated exemplar is an extract of student evidence, with a commentary, to explain key aspects of the standard. It assists teachers to make assessment judgements at the grade boundaries.

New Zealand Qualifications Authority

To support internal assessment

© NZQA 2016
<table>
<thead>
<tr>
<th>Grade Boundary: Low Excellence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.</strong> For Excellence, the student needs to efficiently construct a plan for a basic computer program for a specified task. This involves:</td>
</tr>
<tr>
<td>- constructing a flexible and robust plan</td>
</tr>
<tr>
<td>- specifying an effective procedural structure that constitutes a well-structured logical solution to the task</td>
</tr>
<tr>
<td>- specifying a comprehensive set of test cases with expected, boundary and invalid input for testing the program.</td>
</tr>
</tbody>
</table>

The student has constructed a plan for a basic computer program for a simple Pacman game.

This is an excerpt of the student's overall plan (1) for a basic Pacman program that is flexible (2) and robust (3). The excerpt (4) of the procedural structure constitutes a well-structured logical solution (5) to the task. A comprehensive set of test cases with expected, boundary and invalid input for testing has been provided (6).

For a more secure Excellence, the student could provide more specificity about the name list variable that is being used in the plan and how it will be handled.
Game Variables

| Lives Variable - Stored as a number variable which is displayed on the bottom corner of the screen. Lives starts off at 3, if the player loses all 3 lives they then lose the game. If the user completes the game with more than 0 lives they win the game |
| Name Variable - Stored as a list variable storing text which is displayed at the end of the game along with the users score. The name is asked for at the beginning of the game and is saved until the player wins. No name is displayed on loss. |
| Score Variable – Stored as a number. Used to keep track of the users score as they collect rewards. |

Broadcast

| Ghost starting - When start is clicked the broadcast "Ghostgo" is made which starts the ghost off and gets them moving. |

Assignment of Variables

| At the start of the game lives are set to 3 and the score is set to 0. If the player touches a ghost one life is deducted. Every time the player touches a yellow dot they are rewarded with +10 points. If the player touches the fruit they are rewarded with +50 points. The game is won if the score is at, or exceeds, 840. The game is lost if the players lives equal 0. |

Predefined Actions

| Pacman - Throughout the game pacman switches from costume one to costume two. Costume one is pacman as a yellow circle, costume two is pacman with part of the circle missing to show a mouth. This is on a forever statement which means that throughout the game it looks as if the pacman is opening and closing his mouth continuously, like to real pacman game. |

| Ghosts - During the game if the pacman, controlled by the user, happens to touch one of the four larger dots in the middle then a broadcast is made which changes the ghosts to a new costume, this was added to make the game closer to the real version. NOTE: THE USER CAN NOT EAT THE BLUE GHOSTS! |

User Input

| Pacman - Pacman is controlled by the four arrow keys, each key represents the direction the pacman will go e.g. up key means the pacman will go up. When the arrow key is pressed the directions x or y is changed by 2. e.g. if the pacmann goes left his x is changed by -2. This is to allow for a smoother game. This is also so the pacman doesn't glitch when he hits the walls as when the pacman touches the wall his x or y is just changed by the needed number to make it look as if he is still. |

Input from users

<p>| Name - At the start of the game a bar comes up with the option to type within it. The bar asks the player for their name. The game stores this name in a table variable called Name along with the players score. This table becomes visible to the player if they win the game. It is displayed with sprite 7 which is cued to display if the users win broadcast is received. |</p>
<table>
<thead>
<tr>
<th>Testing</th>
<th>Data for testing expected</th>
<th>Expected test output</th>
<th>Test results correct?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing</td>
<td>“Joe” (entered for name)</td>
<td>Name shown correctly upon winning</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pacman catches yellow dot</td>
<td>Player’s score goes up by 10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pacman catches fruit</td>
<td>Player’s score goes up by 50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pacman touches ghost</td>
<td>Lives left score goes down by 1</td>
<td></td>
</tr>
<tr>
<td>Data for testing boundary</td>
<td>Data for testing boundary</td>
<td>Expected test output</td>
<td>Test results correct?</td>
</tr>
<tr>
<td></td>
<td>Players score gets to 839</td>
<td>Game keeps playing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Players gets to 840, Lives &gt;0</td>
<td>Game Stops, Winning message and players name displayed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Players score gets to 840, Lives =0</td>
<td>Game Stops, game over message displayed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pacman touches ghost, lives left = 1</td>
<td>Game Stops, game over message displayed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Arrow key pressed continuously with in a room</td>
<td>Pacman stops when he comes to walls and turns by 2 degrees each time arrow key pressed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Arrow key pressed continuously when Pacman outside of structures</td>
<td>Pacman does not go off the scene and run around outside of Scratch environment</td>
<td></td>
</tr>
<tr>
<td>Data for testing invalid</td>
<td>Data for testing invalid</td>
<td>Expected test output</td>
<td>Test results correct?</td>
</tr>
<tr>
<td></td>
<td>‘Enter’ key pressed, no other characters entered</td>
<td>Message asking for player to enter name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Any other keys pressed while playing game</td>
<td>No effect to game. Ignored. No errors thrown.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>‘J^e’ entered for name</td>
<td>Message asking for player to enter name using alphabet characters only</td>
<td></td>
</tr>
</tbody>
</table>
2. For Merit, the student needs to skilfully construct a plan for a basic computer program for a specified task. This involves:
   - independently constructing the plan
   - specifying a procedural structure with well-chosen actions, conditions and control structures
   - specifying a set of test cases with expected and boundary inputs for testing the program.

This student has created a plan for a basic computer program for a basic facts checker that verifies the answers for numbers between 0 and 12 for multiplication, division, subtraction and addition.

The student has independently (1) constructed the plan and used well-chosen actions (2), conditions (3) and control structures (4). A set of test cases with expected (5) and boundary (6) inputs for testing has been specified.

To reach Excellence, the student could plan to handle situations when dealing with fractional answers, test for invalid inputs such as negative numbers and characters other than the symbols for multiplication, division, subtraction and addition, and check that the second number is zero when the operation is division.
Purpose of my program.
This will check basic facts answers for numbers from 0-12 for multiplication, division, addition and subtraction. I will be programming it in SmallBasic.

Basic Facts Checker

1. Initialize variables
2. Prompt user for number
3. Is this a valid number?
   - Yes: Valid numbers
   - No: Prompt user for symbol and another number
4. Another problem?
   - Yes: Show result
   - No: End
Breakdown of tasks:
Solve equation part,
For the +, I will check if operation is ‘+’, and then add number 1 and number 2 together, save it as result, print result
For the -, I will check if operation is ‘-‘, and then subtract number 2 from number 1, save it as result, print result
For the *, I will check if operation is ‘*‘, and then multiply number 1 and number 2 together, save it as result, print result
For the /, I will check if operation is ‘/‘, and then divide number 1 by number 2 together, save it as result, print result.
Valid numbers
Check if number entered is >= 0 and <=12, if not, then prompt to reenter valid number.
<table>
<thead>
<tr>
<th>Grade Boundary: Low Merit</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. For Merit, the student needs to skilfully construct a plan for a basic computer program for a specified task. This involves:</td>
</tr>
<tr>
<td>- independently constructing the plan</td>
</tr>
<tr>
<td>- specifying a procedural structure with well-chosen actions, conditions and control structures</td>
</tr>
<tr>
<td>- specifying a set of test cases with expected and boundary inputs for testing the program.</td>
</tr>
<tr>
<td>This student has constructed a plan for a basic computer program for a game program where the player hits a ball around using a paddle.</td>
</tr>
<tr>
<td>The student has independently (1) constructed a plan with a procedural structure specified (2) that shows the order of the well-chosen actions (3), conditions (4) and control structures (5). A set of test cases with expected and boundary (6) inputs for testing the program have been specified (7).</td>
</tr>
<tr>
<td>For a more secure Merit, the student could provide more specificity about the testing for the expected and boundary inputs. For example, the score values could have been included to indicate when the backdrops get swapped.</td>
</tr>
</tbody>
</table>
Super Happy Fun Time

Specifying variables and their data types (number, text)
Score (number): Score goes up by 1 point when ball touches paddle after dropping from top of screen. Goal is to score 10 on each level. Name variable (text).

Showing an order to which events occur:
Press green flag to start
Loads screen to enter game
Button for instructions
Button for play
If instruction button pressed, brings up a new backdrop showing instructions
If play button pressed, brings up game interface with a sprite asking for name.
Name is repeated
Level One starts
When ball touches paddle 10 times, broadcast next level
This is done 5 times (5 levels in total)
If ball touches death zone, broadcast end all
Switches background to you’ve failed
Otherwise, when 5 levels are completed, switches to backdrop stating that you have won

If Statements/Broadcasts:
If score reaches 10, broadcast next level
If ball touches death zone, broadcast end all
If score reaches 50, switch background to win screen

Control:
Paddle controlled by mouse (left to right)

Difficulty:
New level means ball moves 5 paces? Faster each time

Score:
Score goes up 1 point each time ball touches paddle
When score reaches 10, next level appears
When score reaches 50, you have one
Win/Lose:
If score reaches 50, you win
If ball touches death zone, you lose

Bug Test:
If name is repeated correctly
Backdrops DO swap
Score does increase and as it increases at 10 the next level appears

Student was independent throughout the planning process.
Occasionally asked me for help but only after having tried

to research the issue and problem solve on his own.
Frequently helped others with the logic of their plan. Mrs McGregor.
**Grade Boundary: High Achieved**

<table>
<thead>
<tr>
<th>Grade Boundary: High Achieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Exemplar for internal assessment resource Digital Technologies for Achievement Standard 91075

1. Variables:
   - numStudents (number)
   - score (number)
   - name (text)

2. Start

3. Get number of students

4. Score < 50
   - Yes: Display name, grade of NA
   - No: Score >= 50 and < 70

4. Score >= 50 and < 70
   - Yes: Display name, grade of A
   - No: Score >= 70 and < 90

4. Score >= 70 and < 90
   - Yes: Display name, grade of M
   - No: Score >= 90

4. Score >= 90
   - Yes: Display name, grade of E

5. Count <= numStudents
   - Yes: Increase count
   - No: Display message

6. Testing:
   - Test Joe 35 -- should get Joe NA
   - Test Jill 65 -- should get Jill A
   - Test John 75 -- should get John M
   - Test Jack 95 -- should get Jack E

7. End

© NZQA 2016
5. For Achieved, the student needs to construct a plan for a basic computer program for a specified task. This involves:

- specifying variables and their data types
- specifying a procedural structure that combines actions, conditions and control structures
- specifying a set of test cases with expected inputs for testing the program.

This student has constructed a plan for a shooting game for the Scratch programming language.

Variables and data types are specified (1), and a procedural structure (2) has been specified that combines actions (3), conditions (4) and control structures (5). A set of test cases with expected inputs for testing the program has been specified (6).

For a more secure Achieved, the student could provide more specificity for the set test cases, with expected inputs for the program and the procedural structure.
Planning out a program/game in Scratch assessment

**Game Overview**

1. How do you win: **score the most points in the amount of time**
2. How do you lose: **score the least points in the amount of time**
3. How do you play your game: **space bar to shot, arrows to move**
4. How does a player have different levels of success: **the score calculated by kills, deaths, time**
5. How does the game increase in difficulty: **by the experience of the opposite player**
6. List out sprites you have: **player 1, player 2, bullet1, bullet2**

**Section 2**

1. What variables will your game have? what are their data types? *(number/text): 3, deaths is a number, kills is a number, time is a number, username is text*
2. List Broadcast conditions: **when space pressed shoot bullet, when time finishes countdown broadcast winner to player that has the most kills.**
3. For each of the main sprites write out what it will do in a pseudo code format.

```
Players
When key pressed.
    Repeat until not key pressed.
    Point in direction.
    Move 5 steps.
When flag clicked
    Show
    Set player deaths to 0
Forever
    If touching bullet then
    Change player deaths by 1

Bullet
When arrow key pressed
    Hide
    If key pressed then
        Go to player
        Point in direction of player
    Repeat until touching edge or touching player
    Show
    Move 15 steps
    hide
```

4. All rules/conditions the sprite will have assigned to it for complete game play
5. Put all information in order grouped by sprite and order the sprites
Bug Testing

1. List out all testing you will need to do to ensure there is no bugs in your code. **Glitches:** make sure that the player can't go out of the stage, make sure that all the controls work and do what they are meant to do,

2. Be specific when speed increases should happen, score changes, damage everything – if shot +1 death. If shoot and hit +1 kill

3. Make sure that it displays name correctly. ‘Marlene’ entered, should display as ‘Marlene’
<table>
<thead>
<tr>
<th>Grade Boundary: High Not Achieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>6. For Achieved, the student needs to construct a plan for a basic computer program for a specified task. This involves:</td>
</tr>
<tr>
<td>• specifying variables and their data types</td>
</tr>
<tr>
<td>• specifying a procedural structure that combines actions, conditions and control structures</td>
</tr>
<tr>
<td>• specifying a set of test cases with expected inputs for testing the program.</td>
</tr>
</tbody>
</table>

This student has constructed a plan for a shooting game for the Scratch programming language.

The procedural structure (1) shows the actions that will occur, and the conditions (2) and control structures (3) are implied. A set of test cases with expected inputs for testing the program has been specified (4).

To reach Achieved, the student could specify variables and their data types.
Scratch Plan

Game play
The game will be an asteroids version of a paddle game. You will have to get up to a certain score to save the town underneath your paddle. If your score reaches 75 then you win the game. If the asteroids touch the paddle they will bounce off at a random direction. The game has 3 asteroids that will come out at a certain score. If an asteroid touches the town the game is over and you lose.

Paddle coding: When Green flag clicked hide.
When I receive start game forever show set x to mouse x.
When I receive game over hide.

Difficulty
The difficulty will increase by increasing the speed of the asteroids, a maximum of 1, and by adding more asteroids, a maximum of 3. If you're score reaches 10 then the game forever increases the original asteroids speed by 1 of its basic 10. The game will also broadcast Show, this will forever add another asteroid at the starting speed of 10. When the score reaches 30 then the other asteroids speed is also forever increased by another 1. If the score is 30, Show2 is broadcasted and another asteroid is added. And then if the score reaches 50 this asteroids speed is also forever increased by 1.

Broadcasts

At the start of the game an instructions screen and start button are broadcasted. When the start button is clicked by the mouse broadcast “what is your name”. When the player puts in their name broadcast start game. When the score reaches 30 broadcast show, showing another asteroid making two asteroids in the game now. When score reaches 50 broadcast show2, making the third and final asteroid show. When the score reaches 75 broadcast win. If an asteroid touches the town broadcast game over.

Testing scenarios
-I will test if the asteroids forever bounce off of the paddle when they connect.
-I will test if the paddle forever follows the location of the mouse on the screen.
-I will test if when an asteroids touches the town if the game broadcasts game over.
-I will test if when the score reaches 75 that the game broadcasts win.
-I will test that there are no glitches.