Kura SOLUTIONS

# ASSESSMENT

AS91906v1 Level 3 Credits 6

## USE COMPLEX PROGRAMMING TECHNIQUES TO DEVELOP A COMPUTER PROGRAM

| CANDIDATE TO FILL OUT | |
|---|---|
| First and last name | |
| NSN number | |
| Date | |
| Signature | |

## ASSESSMENT RESULT (circle one)

Achieved Merit Excellence Not Achieved

| Resubmission requirements (if any) | |
|---|---|
| Resubmission Date: | |
| Assessor Name | |
| Date Assessed | |
| Signature | |

### ⚠ ASSESSOR NOTE:

Ensure you read all the information in this assessment before you hand it out to learners. The Assessment Schedule and important assessor information are found at the end of this assessment.

# CANDIDATE DECLARATION

## What you need to know

- There is one assessment task that consists of several items.
- You can use your classroom notes or any other relevant resources to help you complete this assessment.
- Make sure your answers are your own work.

### ⚠ YOUR DECLARATION

- I have read and understood the 'What you need to know' above.
- Anything I was unsure of has been explained to me by my assessor.
- I am aware of any requirements, such as health and safety, that I must follow.
- All work completed for this assessment is my own. I did not get help from anyone else.
- I know how to ask for a review of my assessment result.

| Your Signature | Date |
|---|---|
| | |

# Assessment Task One

**Important:** Read through **all** the instructions in this task before you start work on it.

In this assessment, you need to develop a computer program by using complex programming techniques. Your assessor will give you the brief for the computer program you need to develop.

## What you will be assessed on

You will be assessed on how well you do the following.

- Write code for the program to allow it to perform the specified task
- Use complex techniques to develop the program
- Set out the program code
- Document the program
- Test and debug the program

Use the checklist below to ensure you meet all the requirements of the assessment. As you complete an item, tick it off in the 'Student to tick' column.

| Item | Student to tick | Assessor to tick |
|---|---|---|
| 1. Write code for a program to perform the task specified in the brief. The computer program needs to use/do the following. | | |
| • Variables storing at least two different types of data (eg numeric, string/text, Boolean) | | |
| • All of the following control structures:<br>□ Sequence<br>□ Selection<br>□ Iteration | □<br>□<br>□ | □<br>□<br>□ |
| • Get input from a user, file, sensors, or other external source | | |
| • Produce output | | |

Continues next page

| Item | Student to tick | Assessor to tick |
|---|---|---|
| • Use the following complex programming techniques:<br>□ Using a graphic user interface (GUI) that allows the user to interact with the program by inputting values, clicking a button, etc (eg using Pygame)<br>□ Reading from, or writing to, files<br>□ Using third party or non-core API, library, or framework (eg Pygame)<br>□ [OPTIONAL] Object-oriented programming using class(es) and objects you define yourself | □<br>□<br>□<br>□ | □<br>□<br>□<br>□ |
| 2. Ensure the program code is set out clearly. For example, make sure you have used:<br>• indentation correctly<br>• spaces between blocks of code. | | |
| 3. **[MERIT GRADE]** Make sure that the code follows the conventions of the programming language you have used. In other words, you need to stick to the guidelines and commonly accepted ways of doing things for the programing language. | | |
| 4. **[EXCELLENCE GRADE]** Ensure the program is well-structured, and is a logical response to the task specified in the brief. | | |
| 5. **[EXCELLENCE GRADE]** Ensure the program is flexible and robust.<br>Examples of ways to do this include the following.<br>• Use functions, methods, actions, conditions, and control structures effectively<br>• Include input validity checks<br>• Ensure the program correctly handles expected, boundary, and invalid values<br>• Use constants, variables, and derived values – rather than literals | | |
| 6. Document the program with comments. | | |
| 7. **[MERIT GRADE]** Document the program with appropriate names and comments that clearly describe the function of the code and how it behaves (ie what it does). | | |
| 8. Test and debug the program to ensure it works correctly on a sample of expected input. | | |

Continues next page

| Item | Student to tick | Assessor to tick |
|---|---|---|
| 9. **[MERIT GRADE]** Test and debug the program in an **organised way** to make sure it works on both:<br>☐ expected input<br>☐ relevant boundary input. | ☐<br>☐ | ☐<br>☐ |
| 10. **[EXCELLENCE GRADE]** Test and debug the program **comprehensively** to ensure it works correctly for:<br>☐ expected input<br>☐ relevant boundary input<br>☐ invalid input. | ☐<br>☐<br>☐ | ☐<br>☐<br>☐ |

## Evidence to be submitted

You need to submit the following to your assessor.

- The completed program – in electronic format.
- Records of how you tested and debugged the program. This could include a testing plan, annotated screenshots (eg showing the code you have tested and changes you made as a result of testing), a vlog showing how you conducted testing, etc.

Make sure the following are clearly indicated on all evidence you submit:

- Your name
- Your NSN number
- The Achievement Standard number (ie 91906v1).

**...........................................................................END OF ASSESSMENT**

# ASSESSMENT SCHEDULE

AS91906 Level 3, Credits 6

## USE COMPLEX PROGRAMMING TECHNIQUES TO DEVELOP A COMPUTER PROGRAM

**IMPORTANT:** Assessors need to refer to the **Assessment Sample Answers** document and **Efiles** for exemplars that provide an indication of the type and level of response expected for each grade.

| Evidence/Judgements for Achievement | Evidence/Judgements for Achievements with Merit | Evidence/Judgements for Achievement with Excellence |
|---|---|---|
| The learner used complex programming techniques to develop a computer program. This included doing the following. | The learner used complex programming techniques to develop an **informed** computer program. This included doing the following. | The learner used complex programming techniques to develop a **refined** computer program. This included doing the following. |
| □ **Writing code for a program that performs the task specified in the supplied brief [Item 1]**<br>Example:<br>The learner has written a program that is able to perform the task specified in the brief.<br>The program:<br>• is written in a procedural manner, consisting of sequence, selection, and iteration control structures<br>• gets input from a user and external .txt file<br>• uses variables storing different types of data (string – eg score text; numeric – eg values for monster's characteristics; Boolean – True/False; etc)<br>• produces output (eg displays sprites on screen; display high score and player score; etc) | As for Achievement | As for Achievement |

Continues next page

| Evidence/Judgements for Achievement | Evidence/Judgements for Achievements with Merit | Evidence/Judgements for Achievement with Excellence |
|---|---|---|
| • uses the following complex programming techniques:<br>○ Using a graphic user interface (GUI) that allows the user to interact with the program by inputting values, clicking a button, etc – Pygame used to create GUI<br>○ Reading from, or writing to, files (used to save and retrieve high score)<br>○ Using third party or non-core API, library, or framework - Use of Pygame library<br>There may be unnecessary or redundant code. | | |
| □ **Setting out the program code clearly [Item 2]**<br>Example:<br>• The code is indented as appropriate and blank lines are used between blocks of related code and functions. | As for Achievement, **PLUS:**<br>□ **Ensuring that conventions for the chosen programming language are followed [Item 3]**<br>Example:<br>The learner has followed most conventions for the language used for the program (Python).<br>For example:<br>• Variable names are consistently styled (eg lowercase with underscores).<br>• Line length is kept to a maximum of 79 characters for regular code, and 72 characters for block comments and docstrings. | As for Merit, **PLUS:**<br>□ **Ensuring the program is a well-structured, logical response to the task [Item 4]**<br>Example:<br>The program is concise and easily readable, without redundant code.<br>Functions are used rather than duplicating code. |

Continues next page

| Evidence/Judgements for Achievement | Evidence/Judgements for Achievements with Merit | Evidence/Judgements for Achievement with Excellence |
|---|---|---|
| | □ Block comments are used for blocks of code, rather than an inline comment on the first line of the block.<br>□ Block comments are indented to the same level as the block of code they describe.<br>□ Comments are written as full sentences beginning with a capital letter.<br>□ Docstrings are included for each function. These are added after the function definition, are indented, and clearly describe the purpose of the function. | □ **Ensuring the program is flexible and robust [Item 5]**<br>Examples:<br>● Functions, conditions, methods, and control structures are used effectively, with redundant code or unnecessary repetition avoided.<br>● The program includes input validity checks to ensure the program correctly handles expected, boundary, and invalid values.<br>● Where possible, the program uses constants, variables, and derived values, rather than literals. These are set out at the start of the program or relevant functions so that they can be easily found and changed if necessary.<br>● The learner has used objects and classes to improve the flexibility and robustness of the code, and to make the structure of the code clearer. [**Note**: The use of objects and classes is optional. Learners may be able to ensure a flexible program without the use of objects and classes.] |

Continues next page

<table>
<thead>
<tr>
<th>Evidence/Judgements for Achievement</th>
<th>Evidence/Judgements for Achievement with Merit</th>
<th>Evidence/Judgements for Achievement with Excellence</th>
</tr>
</thead>
<tbody>
<tr>
<td>

□ **Documenting the program with comments [Item 6]**

Example:

The learner has included some comments. However, these are not always meaningful and descriptive. Some comments are added to code that is self-explanatory. For example:

```
def update_high_score(score,
high_score):
    """updates scores"""
    if(int(score) > int(high_score)):
      return score
    else:
      return high_score
```

</td>
<td>

As for Achievement, **PLUS:**

□ **Ensuring names are appropriate and comments clearly describe the code function and behaviour [Item 7]**

Example:

The learner has used meaningful variable names which clearly indicate the data held and function names are relevant and appropriate to the purpose of the function.

The comments in the program are clear and meaningful. These comments help to describe the purpose (function) of the code and the code's behaviour. For example:

```
def update_high_score(score, high_score):
    """Checks whether current score is
    greater than high score and,
    if so, updates high score. Otherwise,
    high score is returned as
    updated high score."""
    if(int(score) > int(high_score)):
      return score
    else:
      return high_score
```

</td>
<td>

As for Achievement with Merit

</td>
</tr>
</tbody>
</table>

Continues next page

| | | |
|---|---|---|
| □ **Testing and debugging the program to ensure it works on a sample of expected cases [Item 8]**<br>Example:<br>□ The learner has provided evidence that they have tested the program using expected user input. This was not done in a systematic manner, and the recording of this testing was minimal (eg annotated screenshots of some tests completed, teacher observing the learner as they do testing, etc). | As for Achievement, **PLUS**:<br>□ **Testing and debugging the program in an organised manner to ensure it works on a sample of expected and relevant boundary cases [Item 9]**<br>Example:<br>The learner has used a test plan or some other appropriate method(s) to plan their testing beforehand, and to record the results of this testing. The testing is done in a systematic manner, testing one block of related code (eg a function) for expected and boundary cases (if relevant), before moving on to the next block of code. | As for Achievement with Merit, **PLUS**:<br>□ **Testing and debugging the program comprehensively to ensure it works on expected, boundary, and invalid cases [Item 10]**<br>Example:<br>The learner has systematically tested all input cases, including expected, boundary, and invalid cases. They have methodically conducted and recorded their testing and results. |

# IMPORTANT ASSESSOR INFORMATION

⚠ Refer to the standard specification at www.nzqa.govt.nz.

1. A sample brief for this assessment is provided in the **Assessment Resources** folder. You may need to adjust the brief to ensure it is appropriate for your specific students and context. **IMPORTANT**: Any changes you make to the brief **must** still allow for learners to have sufficient opportunity to meet all requirements of the standard, up to Excellence grade.
2. Sample answers are supplied separately in the **Assessment Sample Answers** document.
3. **This document**, the **assessment sample answers** and any supporting documents (eg brief, resource files, electronic sample answers, etc) must be supplied for external moderation when required.
4. Learner **privacy** and **confidentiality** must be respected always, including for external moderation.
5. **Recording media** (eg video, audio) can be used for verbal responses or as required by the assessment. For more information refer to the NZQA website.