A453 – Controlled Assessment Preparation

Contents

[1. Presentation 2](#_Toc346888931)

[1.1. Cover page 2](#_Toc346888932)

[1.2. Table of contents 2](#_Toc346888933)

[1.3. Headers and footers 2](#_Toc346888934)

[1.4. Styles 2](#_Toc346888935)

[2 The Project 3](#_Toc346888936)

[2.1. Design (9 marks) 3](#_Toc346888937)

[2.1.1. Problem definition 3](#_Toc346888938)

[2.1.2. Proposed solution 3](#_Toc346888939)

[2.1.3. Program design 3](#_Toc346888940)

[2.1.4 Data structures, variables and validation 3](#_Toc346888944)

[2.1.5 Testing and success criteria 4](#_Toc346888945)

[3 Programming Techniques 4](#_Toc346888946)

[4 Efficient use of programming techniques 5](#_Toc346888947)

[5 Development 5](#_Toc346888948)

[6 Testing 6](#_Toc346888949)

# Presentation

## Cover page

You will need a cover page for each task. Each cover page must have the following:

* Centre Number
* Centre Name
* Candidate Number
* Candidate Name
* Unit Code And Title
* Assignment Title

## Table of contents

You must include a table of contents to assist readability. Use styles so that it updates automatically.

## Headers and footers

Footer should contain:

* Candidate Name
* Page Number

Head should contain:

* Centre # 58129
* Candidate # 1111

## Styles

Use Heading styles (Heading 1, Heading 2, …) so that you can have them appear correctly in your table of contents. It’s quite effective to start each main section on a new page. It helps mark to out the flow in the reader’s mind.

Use a special style for your code. It will make it stand out. Courier 10 is good. Like this:

while guess != randomNo:

 if guess <0 or guess>100:

 guess = int(input("Invalid number. Please enter a number between 1-100: "))

 else:

 if guess > randomNo:

 print(guess, " is too high")

 else:

 print(guess, " is too low")

 noOfGuesses = noOfGuesses + 1

 guess = int(input("Please enter another number between 1-100: "))

print("Well done, you guessed correctly. The correct number was: ", randomNo)

print("You guessed correctly in ", noOfGuesses, " guesses.")

You should use the same for pseudocode.

# The Project

## Design (9 marks)



Figure Design Mark Bands

### Problem definition

Present the problem for each task:

* State the problem – what have you been asked to do?
* Make a list of the success criteria. What are the vital parts of the program? You will be referring back to these during testing and evaluation.

### Proposed solution

* List the programming languages/tools/environments you are going to use
* State why you have chosen to use the above
* Will you need to research or learn anything in order to undertake the task?

### Program design

You will need to produce algorithms and/or flowcharts for each of your proposed solutions. You will need to use the standard flowchart symbols. These will need to be detailed for the higher mark bands. You will need to provide evidence of how you have tested your algorithm. For the highest mark band there will need to be some evidence of validation in your algorithms and flowcharts. Subroutines for flowcharts should be included on a separate page.

You may also wish to state any navigation methods, such as menus, and how your program will look (give examples of the menu structure).

* + 1.
		2.
		3.

### Data structures, variables and validation

* State the variables (including name and data types) you will need in your solution
* Describe how any data structures (such as CSV files) will be used in your solution
* Explain any validation that will be put in place (telephone numbers must not exceed 11 characters; gender may only be M or F)

You may wish to document this in table format.

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Name** | **Type** | **Validation** | **Description** |
| userName | String | Must only contain characters a-z/A-Z. | Will be used to store the user’s name. |
|  |  |  |  |
|  |  |  |  |

### Testing and success criteria

This section is about how you are going to test the solutions once it has been coded. It should include bullet points about what your system must do (success criteria), but not how it will do it.

* How will you test your program as you develop it?
* How will you determine whether your final program has been a success?
* Will you get a test-buddy to test the system for you?

It is good practice to use a test plan like the one below. Tests outlined in this table will be undertaken later in the testing section.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Reason** | **Test data** | **Expected Outcome** |
| Check range of user’s guess. | Ensure that the user can only enter numbers between 1-100 in guess the number. | 101 | Program to output “Invalid number. Please enter a number between 1-100”. |
|  |  |  |  |
|  |  |  |  |

# Programming Techniques



Figure 2 Use of programming techniques mark bands

Marks for this section will be awarded from your coded solutions to the individual tasks. Typically, the use of the following features will need to be evidenced:

* Program control; sequence, conditionals (for example IF THEN or CASE) and iteration
* Loops; count and condition controlled (for example repeat and while constructs)
* Data types; string, integer, real and Boolean including string manipulation functions
* File handling; serial files only using open, read, write and close.
* Arrays; limited to single dimensional arrays

# Efficient use of programming techniques



Figure 3 Efficient use of programming techniques mark bands

Again, this section will be assessed through your coded solution. Things to consider for this section: if searching a file will the search stop if the file is found, or will it continue to search until EOF? Would an IF-ELSEIF statement be better than multiple IFs?

# Development



Figure 4 Development mark bands

To be awarded the highest available marks for this section, you will be required to produce an on-going log of your program development. Each time you reach a mini-milestone in the project you will need to produce a screenshot of the code and a brief explanation of how it works. You will also need to provide evidence of systematic testing (screenshots of your code running). For example if you were producing the Guess The Number game, it would be beneficial to produce screenshots and explanations at the following points:

1. Getting the user to input a number between 1-100
2. Validating the number to ensure it is between 1-100
3. Displaying whether the number is too high or too low
4. Allowing the user to guess again if incorrect
5. Counting the number of guesses

As well as documenting each stage you will need to comment your code, explaining the purpose of each section/subroutine.

Your code won’t work first time. Do be sure to details problems you encountered in your development section and how you overcame them.

# Testing



Figure 5 Testing mark bands

In this part of your documentation you will need to complete your test plan from your Design section. For example:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test** | **Reason** | **Test data** | **Expected Outcome** | **Actual Outcome** | **Comments** |
| Check range of user’s guess. | Ensure that the user can only enter numbers between 1-100 for their guess. | 101 | Program to output “Invalid number. Please enter a number between 1-100”. | Program output “invalid number”. | No action necessary.  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

If a test fails you will need to explain what action you took as a result and then re-test you program to ensure the amendment worked. Sometimes it will not be possible to fix the problem in the time scale, so detail how it could be fixed.

In addition to the test plan you should include the following:

* How well your solution meets the success criteria
* Any limitations to your solution (for example, if searching for a date, will it allow you to search for both april, April and 04?)
* How did you develop your solution? Did you complete it all by yourself? Did you problem solve as part of a group? Did you use any library function documentation?
* Did you help any others with their code?