A453 – Controlled Assessment Preparation

Contents

1. Presentation

1.2. 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

1.3. Table of contents

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

1.4. Headers and footers

Footer should contain:

- Candidate Name
- Page Number

Head should contain:

- Centre # 58129
- Candidate # 1111

1.5. 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:
```

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

2.1. Design (9 marks)

Marking Criteria	1-3	4-6	7-9
Design AO2 – 3 AO3 – 6	There will be comments on what the task involves and a limited outline describing the intended approach to some parts of the problem. There will be brief comments on how this might be tested but with no mention of success criteria.	There will be a brief analysis of the tasks indicating what is required for each of the tasks, There will be a set of basic algorithms outlining a solution to most parts of the problem. There will be some discussion of how this will be tested and how this compares to the identified outcomes in the tasks. There will be discussion of the variables to be used and some general discussion of validation.	There will be a detailed analysis of what is required for these tasks justifying their approach to the solution. There will be a full set of detailed algorithms representing a solution to each part of the problem. There will be detailed discussion of testing and success criteria. The variables and structures will be identified together with any validation required.

Figure 1 Design Mark Bands

2.1.2. 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.

2.1.3. 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?

2.1.4. 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).

2.1.5 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.

1.1.2 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 or system to make sure it?

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

	Guidance		
Use of programming techniques	There is an attempt to solve parts of the tasks using few of the techniques identified. [0 - 2]	There is an attempt at most parts of the tasks using several techniques. [3 - 4]	There is an attempt to solve all of the tasks using most of the techniques listed. [5 - 6]

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

Efficient use of programming techniques	The techniques used may not be entirely appropriate to the problem and will only produce partially working solutions to a small part of the problem. [0 - 4]	The techniques will be used appropriately giving working solutions to most of the parts of the problem. Some sections of the solution will be inefficiently coded. [5 - 8]	The techniques are used appropriately in all cases giving an efficient, working solution for all parts of the problem. [9 - 12]

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

ent	There will be some evidence to show a solution to part of the problem with some evidence to show that it works. Code will be presented with little or no annotation, the	There will be evidence to show how the solutions were developed. There will be some evidence of testing during development showing that many parts of the	There will be detailed evidence showing development of the solution with evidence of systematic testing during development to show that all parts work as required.

Development	show a solution to part of the problem with some evidence to show that it works. Code will be presented with little or no annotation, the variable names not reflecting their purpose and with little organisation or structure. [0 - 3]	how the solutions were developed. There will be some evidence of testing during development showing that many parts of the solution work. The code will be organised with sensible variable names and with some annotation indicating what sections of the code does. [4 – 6]	showing development of the solution with evidence of systematic testing during development to show that all parts work as required. The code will be well organised with meaningful variable names and detailed annotation indicating the function of each section. [7- 9]

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

Testing	There will be evidence to show that the system has been tested for function but the test plan will be limited in scope with little structure. There will be little or no evidence to show how the result matches the original criteria. The evidence of written communication is limited with little or no use of specialist terms. Errors in spelling, punctuation and grammar may be intrusive. Information may be ambiguous or disorganised. There will be some comments on others' and their own input into group work. [0 - 3]	There will be a test plan covering many parts of the problem with some suggested test data. There will be evidence that the system has been tested using this data. There will be some evidence to show how the results of testing have been compared to the original criteria. There will be a brief discussion of how successful or otherwise the solutions are. Produces evidence of good written communication using some specialist terms. There will be few errors in spelling, grammar and punctuation. Information for the most part will be presented in a structured format. They will have commented on their own and others' contribution to any group work and [4 - 6]	The test plan will cover all major success criteria for the original problem with evidence to show how each of these criteria have been met, or if they have not been met, how the issue might be resolved. There will be a full evaluation of the final solution against the success criteria. A high level of written communication will be obvious throughout the task and specialist terms/technology with accurate use of spelling will have been used. Grammar and punctuation will be used correctly and information will be presented in a coherent and structured format. They will provide an evaluation on theirs and others' contribution to any group activities. [7 - 9]

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?