Unit Number: AS 91373 – 2.46
Unit Name: Construct an advanced computer program for a specified task
(Version 4)
Level 2
Credits 3

Planned Review Date: 31 December 2018

Learner's Name ……

Instructions

- Read all instructions carefully
- Complete **ALL** tasks
- Ensure your name is on every page of your assessment
- Check that this assessment has pages 1-5 in the correct order and that none of these pages is blank.

Conditions

- The timeframe for completion is Term 2, Weeks 3-5. **Due Date: 1 June 2018**
- You must follow legal ethical and moral responsibilities when creating your documents

I give permission for my assessment to be given to the external assessor for marking

Student:

Learner's Statement of Authenticity
This assessment has been done entirely by me.

Signed Date

Achievement Criteria

Achievement □	Achievement with Merit □	Achievement with Excellence □
Construct an advanced computer program for a specified task. By…	**Skilfully** construct an advanced computer program for a specified task. By…	**Efficiently** construct an advanced computer program for a specified task. By…
● Implementing a plan for an advanced program in a suitable programming language	● **Independently** implementing a plan for an advanced program in a suitable programming language **that uses well-chosen scopes for variables, and well-chosen parameters for modules**	● Constructing an advanced program where the modules **(including their procedural structures) constitute a well-structured logical decomposition of the task** ● **Using variables, constants, and derived values effectively so as to increase the flexibility and robustness of the program**
● Setting out the program code clearly and documenting the program with comments	● Documenting the program with **variable and module names** and comments **that accurately describe code function and behaviour**	● Setting out the program code concisely and documenting the program with comments that explain **and justify decisions**
● Testing and debugging the program to ensure it works on a sample of expected input cases.	● Testing and debugging the program **in an organised way** to ensure it works on inputs **that include** both expected and **boundary cases.**	● **Comprehensively** testing and debugging the program in an organised **and time effective way** to ensure the program is correct on expected, boundary **and invalid inputs.**

SN3	It is preferable for the programming language for this standard to be text-based. Any language chosen must support indexed data structures, modules with parameters, global and local scope of variables and good comment/document facilities.
SN4	An advanced computer program must have a modular structure, an indexed data structure (e.g. array or list), input and output, and procedural structures that combine sequential, conditional and iterative structures.
SN5	A program with a modular structure contains a collection of named modules (procedures, functions, methods, or subroutines) where each module implements a procedural structure for a sub-task. At least the top-level module (and possibly others) must contain calls to other modules. The modules should include parameters as needed.
SN6	A specified task refers to a set task which requires the development of an advanced computer program to resolve. The task must be of sufficient rigour to allow the student to meet the standard and needs to be agreed prior to the program being constructed. It may be teacher-given or developed in negotiation with the student. It is expected that most students constructing a program for this standard will have developed a plan for an advanced computer program for this task. If this is not the case, the teacher should provide the student with an abstract plan to guide their program development but one which cannot be directly transcribed to the programming language.
SN7	The scope of a variable may be global (accessible from all modules) or limited to a single module. A well-chosen scope matches the way the variable is used.
SN8	Well-chosen parameters for modules are those where the number and types of parameters are decided in the context of the module's sub-task.
SN9	Constants should be used as required when a value never changes. Derived values are returned properties or are calculated from other values. Examples include but are not limited to: the length of an array or string; area which is calculated from the width and height of a rectangle; and the mid-point of a graphics object which is calculated from its width and height.
SN10	In a well-structured logical decomposition of the task, each module will have a clear and well defined purpose within the context of the task. Interaction between modules will be minimised, modules will be reused rather than duplicated, and the procedural structure of each module will be efficient.

MOST IMPORTANT NOTE

With this assessment you may open and use any previous program you have created plus any notes that YOU have written yourself

You are not allowed to get help from the Internet, reference manuals or other students

You can get limited help from your teacher if you need it, but if you ask for help you will no longer be able to gain Merit or Excellence

Other Notes

- It is suggested that you follow each of the instructions in order to ensure you meet the requirements to pass the assessment
- These instructions will assume you know what you are doing. For example, if you get an instruction 'test the program is working correctly' it is assumed you know how to!

Assessment Task

Ruakaka Pizzas want to computerise their phone orders. Specifically, they want to be able to enter customer details, pizza(s) ordered and pick-up or delivery requirements into a computer and have it display the delivery details, itemised order, and total cost. Phone orders generally consist of several kinds of pizza.

Your program must meet the following specifications:

- The program contains options for the phone operator to specify whether the pizza order is for pickup or delivery.
- If the order is for delivery:
  - the program should collect the customer's name, address and phone number
  - a $3 delivery charge should be added to the total cost
- If the order is for pick up:
  - The phone operator should be asked to enter the customer's name into the program.
- The program should allow the phone operator to input how many pizzas the customer would like (maximum 5).
- A numbered menu of at least 12 pizza names should be presented to the phone operator. This menu should be stored in an indexed data structure (e.g. an array or list) and may be hard-coded within the program so that pizza names don't need to be typed in every time the program is run.
- Each pizza to be ordered should be selected from the choices available on the menu and the order information should be stored.
- The cost of the first seven (regular) pizzas on the menu is $8.50 and the rest are $5 more as they are gourmet pizzas.
- When the order is finished:
  - the names of ordered pizzas and their individual price should be displayed
  - the total cost of the order including any delivery charge should be displayed
  - customer name should be displayed
  - if the pizza is for delivery the address and phone number should be displayed.
- The program should allow the operator to cancel the order.
- After the order information has been displayed the program should be ready to accept another order or exit.

Note: There does not need to be any facility for editing the ordered list of pizzas. If the order is not right, the operator would just cancel it and start again from the beginning.

Ensure that in your finished program:

- there is at least one indexed data structure and is decomposed into user-defined modules with scoped variables, constants and derived values effectively to increase the flexibility and robustness of the program.
- interaction between modules is minimised, modules are reused rather than duplicated, and the procedural structure of each module is efficient.

Annotate your code and use explanatory variable/module names so that the purpose of each part of the program is clear. This includes documenting the program with comments on the function and behaviour of the modules. Use your comments to justify the decisions you made during the programming process.

Comprehensively test and debug the program in an organised and time effective way so it works correctly on all inputs – expected, boundary, and invalid.

Hand in:

- a printout of your final source code
- an electronic version of your program
- screenshots of example output with annotations
- short documentation describing how you implemented your testing procedures and the outcomes of your testing e.g. a testing log.

Task 1

Place comments at the start of your program that describe the purpose of the program □

- As you continue writing your program add suitable comments

Task 2

Create a list or array that stores the 12 types of pizza. □

Task 3

Create code, possibly in a subroutine, that asks price of the pizza □

- It is recommended this is done using a loop
- It is recommended that these values are stored in a list or array

Task 4

Write the code that will ask the phone operator to determine if the order is for pick up or delivery. If pick up it will ask for customer's name. If deliver, it will ask the phone operator to enter the customer's name, address and phone number.

Task 5

Create a subroutine that will calculate the cost of the order including delivery if necessary □

- It is a **requirement** that a subroutine is used for this task
- It is recommended this is done using a loop
- It is recommended that these values are stored in a list or array
- Make use of global variables if required by the language you are using
- You can include task 3 in the same subroutine or loop if you wish

Task 6

Write the code, possibly in a subroutine, that will print the required output □

- It is recommended this is done using a loop

Task 7
Write the code that uses the subroutines, plus add any other necessary code □

Task 8
Check the program for robustness □
- Try to make the program fail by putting in strange inputs
- Getting this to work properly is only required for Excellence

Task 9
Check the program for boundary variables □
- Put inputs in that are on the boundary of different saver types
- Getting this to work properly is only required for Merit

Task 10
Create a testing table □
- In Word or a similar application create a table that contains a range of possible inputs, the expected outputs and if this was achieved with testing
- Doing this is not a requirement, but is recommended

Task 11
Printouts □
- Print out your code, outputs and if you did it the testing table

PROGRAMMING CHECKLIST

To pass this assessment it is necessary for you to have met certain requirements
Before you hand in your assessment for marking ensure you have done ALL of the following:

Requirements for Achieved

In order to gain an Achieved in this assessment you MUST

- Produce a program that works as per the specifications…
    - It must ask for the required information
    - Print out in a format suitable for the phone operator to be able to place the order
- Set the program out clearly
- Use variable names that match their function in the program
- Include reasonable comments throughout the program
- Ensure the program has been tested and fixed so it works with expected inputs
- Have at least one array or list
- Get input and have output
- Have conditional statements such as IF
- Have at least one loop such as WHILE or DO UNTIL
- Have modules or subroutines for the price calculation
- Use a module or subroutine name that describes what its function is
- Have different sections to do different parts – not just have subroutines
- Have at least one constant – such as the number of times to go around a loop

For achieved, limited guidance from your teacher is allowed

Requirements for Merit

You MUST meet all requirements for Achieved, PLUS…

- Work independently – no help from your teacher is permitted
    - You may refer to existing programs you have written and your notes
- Choose the appropriate scope for the variables. Such as choosing global or local variables
- Have well-chosen parameters for the modules (such as subroutines)
- Have sufficient comments in the program to describe its function and what each section does
- Test the program
    - In an organised way (such as using a checking table)
    - That it works correctly with expected and boundary inputs
    - And have fixed any problems found during testing and re-tested
    - And provide printed evidence of this testing

Requirements for Excellence

You MUST meet all requirements for Achieved/Merit, PLUS…

- Your program must be well structured and be laid out logically
- Have modules/subroutines where they should reasonably be used
- Use variables, constants and derived variables effectively and robustly
- The program must be concise – be efficient with the amount of code required
- Have comments that explain what each step is doing and justify why
    - The number of lines of comments will not be counted against the need for efficient code
- Fully test the program including…
    - Testing it for time efficiency
    - Unexpected variables, such as
        - Text where numbers are expected and dealing with it
        - Orders that are not reasonable
        - Your code should deal with unexpected inputs and take reasonable action
    - Producing a short report explaining how testing was carried out