## Specs:

12 types of fish:

* 6 cheaper: $4.10 → Shark, Flounder, Cod, Gurnet (two you can add),
* 6 deluxe & more expensive: $7.20 → Snapper, Pink Salmon, Tuna, Smoked Marlin (two you can add)

Program specs:

* The programme contains **options** for the phone operator to specify whether the Fish order is **Cooked or Frozen** and for **Pickup or Delivery**.
* If frozen: The program should apply a discount of $1.05 per fish item.
* If delivery: the programme should collect the **customer’s name, address and phone number**
* A **$5 delivery charge** should be added to the total cost.

**Introduction/Kupu Arataki**

This assessment activity requires you to develop and test a computer program to process phone orders for Frozen or cooked Fish takeaways, using advanced programming techniques.

You will be assessed on how effectively you refine your program to ensure that the program:

· is a well-structured, logical response to the task

· has been comprehensively tested and debugged

· is flexible and robust.

You should consider these examples of ways of making a program flexible and robust:

· using actions, conditions, control structures and methods, functions, or procedures effectively

· checking input data for validity

· correctly handling expected, boundary and invalid cases

· using constants, variables and derived values in place of literals.

When developing your program, you must ensure your program:

· uses variables storing at least two types of data (e.g. numeric, text, Boolean)

· uses sequence, selection and iteration control structures

· takes input from a user, sensor(s), or other external source(s)

· produces output

· follows common conventions of the chosen programming language

· is documented with appropriate variable/module names and comments that describe the code function and behaviour.

Your program must use these advanced programming techniques:

· modifies data stored in collections (e.g. lists, arrays, dictionaries)

· defines and manipulates multidimensional data in collections

· creates methods, functions, or procedures that use parameters and/or return values

**Task/Hei Mahi**

**Scenario:**

Freddy's Fast Fish wants to computerise their phone orders. Specifically, they want to be able to enter the Customer Details, Fish and Chip Orders, Cooked or Frozen, for Pick-up or Delivery options into a computer program and have it process and display the Delivery Details, Itemised Order, and Total Cost.

Phone orders generally consist of several kinds of Fish items.

There are 12 types of fish, Six Cheaper Fish Types $4.10 Each (Shark, Flounder, Cod, Gurnet (two you can add)) and Six Deluxe Fish Types $7.20 Each (Snapper, Pink Salmon, Tuna, Smoked Marlin (two you can add)).

· The programme contains **options** for the phone operator to specify whether the Fish order is **Cooked or Frozen** and for **Pickup or Delivery**.

· If the order is Frozen:

o The program should apply a discount of $1.05 per fish item.

· If the order is for **Delivery:**

o the programme should collect the **customer’s name, address and phone number**

o A **$5 delivery charge** should be added to the total cost.

· If the order is for **pick up:**

o The phone operator should be asked to enter the **customer’s name and phone number** into the programme.

· The programme should allow the phone operator **to input** **how many items of Fish** the customer **would like** (***maximum of 7 per fish type***).

o The program should display a nice error if there is more than 7 of one item.

· A **numbered menu** of **All Fish Items** should be presented to the phone operator. This menu should be stored in an **indexed data structure** (e.g. an array, list, or multidimensional array) and **may be hard-coded** within the programme **so that Fish names don’t need to be typed in every time the programme is run**.

· Each Fish item to be ordered should be **selected from the choices available** on the menu and the order information should be stored.

· When the order is finished:

o the **names o**f **ordered Fish Items** and **their individual price** should be **displayed**

o The **Total Cost** of the order including any **Delivery Charge** should be **Displayed**

o **Customer Name** should be **Displayed**

o If the Order is to be **Cooked** or **Frozen**

o If the order is to contain Cooked chips and the value of the Chips.

o If the order is **for Delivery** the **Address and Phone Number** should be **displayed.**

· The programme should allow the **operator to cancel the order.**

· **After the order** information has been **displayed** the **programme** should **display** the optionto **accept *another order*** *or* ***exit.***

**You need to think about:**

· How will you program and present the menu and receive user input?

Will you give options of which fish and quantity or ask quantity for every fish?

· How will you store your data?

What variables will you require and what type of data will your variables store (e.g. text, numeric, Boolean)? Will you store data in collections (e.g. lists, arrays or dictionaries) to improve the structure, flexibility and robustness of your program?

· How will you structure your program?

What procedural structure will your program require? Will you create functions/method/procedures to improve the structure, flexibility and robustness of your program? What parameters and/or return values would be required?

· How will you validate input and give feedback to the user?

What methods will you use to restrict and/or validate input. When will the program display output to the user?

· How will you document your error/bug checking?

will you use screen shots and annotate them, will you make a short video…?

**Development:**

· You should break the program up into components. Think about what information each component will need to do its job, and what information it will pass on to the rest of the program. Code, test and debug each component separately. As you complete each section you should save your code with a new version number.

**Note:** To test a program in a comprehensive way, you should think about how you will test the program for various cases such as expected, boundary and unexpected input. It is often useful to note down what you want to test and what you expect to happen, as well as what actually happened. Testing can be demonstrated by making a brief screencast showing the program being comprehensively tested. If desired, you can take screenshots of your screencast and annotate them. [Print Screen] and [Alt] + [Print Screen] will help you show your testing.

· Ensure that you comment your code appropriately as you develop it and use variable/module names and comments that describe code function and behaviour.

· Ensure that you have followed conventions for the programming language of your choice and that your program is a well-structured, logical response to the task.

· You should ensure that your code is robust and that it handles expected, boundary and invalid cases.

· Wherever possible you should try to ensure that your code has a flexible structure to allow for continued development.

**Notes:**

· Cooked or frozen is for the whole order or for each fish type, you can choose.

· Max of 7 means per fish type not quantity for total order i.e. they can order all 12 types, with a max of 7 of each, so a total quantity of 84 fish items.

· Chips, can be in scoops (and you give a value how much a scoop is worth) or in Dollars and they just say how many dollars.

i.e. either SCOOPS - 1.5 scoops of chips (which you convert to dollars) or DOLLAR - $6 of chips.

Testing new keyboard it seems to be nice all thought i don't really know when im typing as it seems to not have any clicks but as I get used to it it seems to be nice but you do need to push down quite hard, but I can get used to that seeing as this key board was designed to be quite therefore no mechanical I say it's pretty good I might actually love it. Now I need to get the screen in the right spot and hang one. I have changed the tilt of the screen and typing this seems to be easier. The tilt was good now let me change the chair height. that ‘s good now Ive got the spot this is good. Rudy out.