## dComputer Program Design Plan

|  |  |  |  |
| --- | --- | --- | --- |
| * + 1. **Program function** | | | |
| *Indicate the function/purpose of the computer program (ie what does it need to be able to do and why).* | | | |
| The purpose of this program is to help a **mobile Donut start-up business owner**to calculate the costs of his work so he can invoice his customers accurately. He has three types of donuts, for which he charges different prices (single donut prices only – no box sets available):   1. Chocolate - $2.50 2. Caramel - $2.50 3. Cinnamon - $1.20   He will charge a flat fee of $5 for delivery which covers 20km radius from central location of bakery. Minimum donuts are set to 5 and maximum is 30. The client works on weekdays and orders can be taken in the weekend. He wants 3 working days to process and deliver the donuts. Client does work on weekends.  This program allows him to first choose the type of donut, then input the number of donuts for each type. It will then calculate and display:   1. the total no. of donuts per donut type 2. the total cost for each type of donut 3. the grand total costto charge the customers.   *Key features of the program will be:*   1. date format set to long date e.g. Monday, 25 March 2019 2. customer details will include the following:   Customer details   * 1. Customer Order Number for each day ,   2. First and Last name,   3. phone Number (to confirm location before delivery),   4. delivery date (format as short date e.g. DD/MM/YY)  1. Client wants the following details to appear for each customer per date in txt file    1. Customer Order Number:    2. Date of Order:    3. First Name:    4. Last Name:    5. Delivery Date:    6. Customer Phone Number:    7. Order Details: e.g. 5 chocolate, 2 caramel, 3 cinnamon    8. Total Purchase Cost: $   *Other features could include the following:* | | | |
| * + 1. **Target language** | | | |
| *Indicate the target programming language (e.g. Python 3 incl. version) for the computer program.* | | | |
| *Python 3.5.1 or 3.4* | | | |
| * + 1. **Data dictionary** | | | |
| *Indicate the data items, data types, scope (local/global) and the variables/constants (incl. functions/modules, lists, derived values). An example is given under the column headings.*  *Remember:*   * *you need to create 2 or more user-defined functions/modules (include: comments to explain the purpose of each user-defined function/modules)* * *any modules you create needs to be correctly named, and saved as py file* | | | |
| **Description of Data item**  ***Example***:  *User’s age (i.e. get\_userage function is used to ask for user’s age and store it in a local variable called user\_age for use when the function is called)* | **Data Type**  *Number - integer* | **Variable Scope (local / global)**  *local* | **Variable/Constant (incl. functions/modules, lists, derived values)**  *user\_age* |
| **Global constant that contains the price for chocolate donut e.g. 2.50** | **float** | **global** | **CHOCOLATE\_PRICE** |
| **Global constant that contains the price for caramel donut e.g. 2.50** | **float** | **global** | **CARAMEL\_PRICE** |
| **Global constant that contains the price for pineapple donut e.g. 3.50** | **float** | **global** | **CINNAMON\_PRICE** |
| A flat fee of $5 for delivery which covers 20km radius from central location of bakery will be charged | **float** | **global** | **TRAVEL\_FEE** |
| *User’s first name and last name (i.e. get\_username function is used to ask for user’s first and last and stores them in local variables called fname and lname for use when the function is called)* | **str** | **local** | **fname** |
| **str** | **local** | **lname** |
| **A list of types of donuts available for sale e.g. choco….** | **str** | **global** | **donut\_list** |
| Minimum donuts are set to 6. | **Int** | **global** | **MIN\_DONUT** |
| maximum donuts are set to 30. | **int** | **global** | **MAX\_DONUT** |
| *Type of donut to order (i.e. get\_donut\_typefunction is used to ask the user to select the type of donut from the list of donuts and stores it in local variable called donut\_type for use when the function is called)* | **str** | **local** | **donut\_type** |
| *Get number of donuts per type of donut (positive integer) (i.e. get\_donut\_order function is used to ask the user to enter the number of donuts per donut type and stores it in a local variable called donut\_order. Loop used to check total\_orderis within min and max range (use MIN\_DONUT, MAX\_DONUT constants to track the order range))* | **int** | **local** | **donut\_order** |
| **int** | **local** | **total\_order** |
| *Get the cost of each donut type and store in total cost.* | **Float** | **Local** | **cost\_donut\_type** |
| **float** | **Local** | **total\_cost** |
|  |  |  | **delivery\_date** |
|  |  |  |  |
|  |  |  |  |
| * + 1. **Algorithmic structure (Procedural / Functions & Methods / Objects & Classes)** | | | |
| *Indicate the algorithm for the program using a suitable algorithm tool(s). For more information, see notes.*  ***Remember:****You must use a combination of pseudocode and decomposition diagrams.* | | | |
| *Hyperlink all versions of pseudocode (comment errors found and fixed for each version)*  *Hyperlink all versions of the program code (comment errors found and fixed for each version)* | | | |
| **Algorithmic structure** | | | |
| *decomposition diagrams – shows the main routine of program (final version of program)* | | | |

|  |
| --- |
| * + 1. **Testing procedures** |
| *Indicate test cases for testing the planned program.* |
| |  |  |  |  | | --- | --- | --- | --- | | **Test:** | **Expected Result:**  **Refers toAlgorithm (incl. pseudocode)** | **Actual Result:**  **Refers to program**  (see screenshot section below) | **Result: Pass / Fail?**  **Incl. comments relating to “find and fix” errors** | | **EXPECTED / NORMAL INPUTs** | | | | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | **BOUNDARY / EXTREME INPUTs** | | | | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | **EXCEPTIONAL / INVALID INPUTs** | | | | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  |   **Screenshots of Testing:**  ***Remember*:**   * *Clearly label each screenshot (e.g. figure 1); and* * *State which test caseeach screenshot relates to in your testing table* |

|  |
| --- |
| 1. **Project Plan** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Progress Plan (18741)**  *You may use this “Progress plan” table to help manage your time, resources keep discussion notes (client) and document steps you took to troubleshoot issues/problems you encountered.*  *You can hyperlink files to show evidence of meetings with client, troubleshooting issues with pseudocode (incl. versions) and program(incl. versions)* | | | | |
| **Resources Required (incl. software and hardware):** | | | | |
| ***Date and Time***  ***(incl. minutes/period):*** | ***What I did?*** | ***State any issues/problems:*** | ***What I did to solve them?*** | ***Discussion notes (Client meetings)*** |
| **18/3/19 – 22/3/19** | **Brief log and initial client meeting to check and get specs clarified** |  |  | **22/3/19 – meeting with client – see meeting notes**  [**[Brief and Meeting Notes]**](Brief%20and%20Log.docx) |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **4/4/19 – LAST Day (Due today)** |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **Back log cleared by** |  |  |  |  |

|  |
| --- |
| 1. **End-User Guide** |
| *The End-User Guide must include the following:*   1. *The Purpose or description of the program* 2. *Instructions on how to access the program* 3. *Instructions on how to use the program* |