**Define the problem**

**Task Description**

A local charity organisation is running a fundraising event based on the TV show “Deal or No Deal”. Contestants pay a set fee to have the chance of being either a contestant or a podium player. Your software company has been asked to develop a program that will allow players to participate in a virtual version of the game.

**RULES OF THE GAME**

The Contestant selects one briefcase from 26. The 25 remaining briefcases are given to the 25 Podium Players, who take turns in opening their numbered cases, in the order decided by the Contestant. 25 of the cases contain an insert that represents a cash value.

As each case is opened, its contents are revealed to the audience, and the Contestant. The cash values contained in the opened cases are no longer available to be won. Whatever is represented inside the Contestants briefcase at the end of the game is what the Contestant will win, unless they accept an Offer from the Bank during the course of the Game.

**THE BANK**

During the game the contestant is made various offers by the Bank to sell their (the Contestants) briefcase for a cash sum. If the Contestant accepts a Bank Offer, they win the cash sum offered by the Bank, and are no longer entitled to the sum that is eventually revealed to be in the Contestants briefcase.

**OPENING THE BRIEFCASES**

The 26 briefcases are opened according to the following system:

6 cases are opened. The Bank makes an offer based on the remaining 20 closed cases. The Contestant decides whether to accept or reject the offer - Deal or No Deal.

5 cases are opened. The Bank makes an offer based on the remaining 15 closed cases. The Contestant decides Deal or No Deal.

4 cases are opened. The Bank makes an offer based on the remaining 11 closed cases. The Contestant decides Deal or No Deal.

3 cases are opened. The Bank makes an offer based on the remaining 8 closed cases. The Contestant decides Deal or No Deal.

2 cases are opened. The Bank makes an offer based on the remaining 6 closed cases. The Contestant decides Deal or No Deal.

The remaining unopened briefcases are then opened one at a time. In this portion of the Game, one Bank Offer is made to the Contestant after each individual case is opened. **If the Contestant opens their selected case before the programs Host or the Producer directs them to do so, the Game will halt immediately, the Contestant will be eliminated and the Game will be declared null and void.** The Contestant will not be eligible to take part in any further Games.

**HOW THE BANK MAKES IT OFFER**

All the remaining case totals are added together and then averaged. This amount is then reduced by 15%

**Case Totals**

|  |  |
| --- | --- |
| **.50c** | **$1000** |
| **$1** | **$2000** |
| **$2** | **$3000** |
| **$5** | **$4000** |
| **$10** | **$5000** |
| **$20** | **$10,000** |
| **$50** | **$15,000** |
| **$100** | **$20,000** |
| **$150** | **$30,000** |
| **$200** | **$50,000** |
| **$250** | **$75,000** |
| **$500** | **$100,000** |
| **$750** | **$200,000** |

**IPO**

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| <Form load>;  List Value; | Randomize the value of the cases by randomly select values from list Value. | <Nothing> |
| Select one case (when handleSelection() is called);  List Cases; | Locks case by storing the value in the case to a new global variable “Selected” and then set the value of the case in the list Cases to 0. | Global Variable Selected; |
| <Button press> (Selecting other cases);  List Cases;  List BankTime; | If the status is selection, then call handleSelection();  If the status is open, then call handleOpen();  If the status is finish, do nothing;  If the status is final, call handleFinal(); | Output depends on each functions. |
| Function bank() is called;  List BankTime;  Input from message box | Determine the value of bank offer by calling function calcOffer(), and then show up a message box with yes and no.  If yes is pressed, then the game finishes and the user gets that money.  If no is pressed, then the game continues. | Output varies:  YES – The value that the user can get  NO – Nothing, redirection to updateStatus() |
| Function calcOffer() is called;  List Cases; | Loop through everything in the Cases, determine the number of non-negative values and the sum of all non-negative values, then find the bank offer by the average (sum / number) multiplied by 0.85. | Float bankOffer; |
| Function updateState() is called;  List Cases;  Variable status; | Check the status and sets the new status accordingly.  Before all those, call updateLabel().  If status is load, then instruct user to select one to keep and update state to selection;  If status is selection, then instruct user to select ones to open and update state to open;  If status is open, then loop through the list Cases and determine the number of unopened, unselected case. If that number is in list BankTime, then switch to bank(). Otherwise, do nothing;  If status is bankfinish, then update status to open;  If status is finish, display a message box that tells the user the game is already finished;  If status is final, update status to finish. | Updated status; |
| Choose one to reveal value (Function handleOpen() is called) | Check the value in the case.  If it is a negative value, show up a message box that indicates that case is already open.  If it is 0, show up a message box that the participant is already eliminated, and end the game immediately.  If it is a positive value, set its value to negative (by multiplying it by -1), then update the label for newest value.  After all those, call function updateState() | Revealed value |
| Last round, choose one to open and keep (Function handleFinal() is called);  List Cases; | Check if the value is valid.  Open the case, sets its value to negative.  Call updateState() | Revealed value |
| updateLabel() is called;  List Cases; | Updates the label according to the list Cases | Updated label value. |