**ORNAMENTAL / DISPLAY FUNCTIONS**

*// S = dollar sign*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FUNCTION:** void displayFloatingIcons(int nStart, int nEnd, char cIcon, int nSpacing)  **DESCRIPTION:** prints the cIcon every nSpacing from nStart to nEnd | | | | | |
|  | **TEST DESCRIPTION** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| 1 | Prints cIcon continuously without spaces between characters | nStart = 1nEnd = 20  cIcon = 36(Dollar sign)  nSpacing = 1 | N/A |  | P |
|  | Prints cIcon continuously with 4 spaces between characters | nStart = 0  nEnd = 20  cIcon = 36(Dollar sign)  nSpacing = 5 | N/A |  | P |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FUNCTION:** void displayHeaderUnderline()  **DESCRIPTION:** displays the header underline of each menu | | | | | |
|  | **TEST DESCRIPTION** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| 1 | N/A | N/A | N/A |  | P |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FUNCTION:** void displayLetterScanDivider()  **DESCRIPTION:** displays a divider at the bottom part of the game when scanning a letter | | | | | |
|  | **TEST DESCRIPTION** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| 1 | N/A | N/A | N/A |  | P |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FUNCTION:** void displayAmountScanDivider()  **DESCRIPTION:** displays a divider at the bottom part of the game when scanning numbers | | | | | |
|  | **TEST DESCRIPTION** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| 1 | N/A | N/A | N/A |  | P |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FUNCTION:** void displayUpperUniversalChoiceBox() and void displayLowerUniversalChoiceBox()  **DESCRIPTION:** displays the upper lower part of choice boxes in the game | | | | | |
|  | **TEST DESCRIPTION** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| 1 | N/A | N/A | N/A |  | P |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FUNCTION:** void displayMoney(int nTypeOfMoney)  **DESCRIPTION:** displays the amout of money(nTypeOfMoney) | | | | | |
|  | **TEST DESCRIPTION** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| 1 | nTypeOfMoney is less than 1000 | nTypeOfMoney = 974 | 974 | 974 | P |
| 2 | nTypeOfMoney is greater than or equal to 1000, and displays amount with comma | nTypeOfMoney = 1034 | 1,034 | 1,034 | P |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FUNCTION:** void displayPrice(int nPriceOfProduct)  **DESCRIPTION:** displays the price(nPriceOfProduct) of essential and diffuser oils per bottle | | | | | |
|  | **TEST DESCRIPTION** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| 1 | nPriceOfProduct is less than 1000 | nPriceOfProduct = 304 | 304/btl | 304/btl | P |
| 2 | nPriceOfProduct is greater than or equal to 1000, and displays amount with comma | nPriceOfProduct = 1302 | 1,302/btl | 1,302/btl | P |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FUNCTION:** void displayDropPrice(int nProductPrice)  **DESCRIPTION:** displays the price(nProductPrice) of essential and diffuser oils per drop | | | | | |
|  | **TEST DESCRIPTION** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| 1 | nProductPrice is less than 1000 | nProductPrice = 546 | 546/ drp | S 546/drp | P |
| 2 | nProductPrice is greater than or equal to 1000, and displays amount with comma | nProductPrice = 1234 | 1,234/ drp | S 1,234/ drp | P |

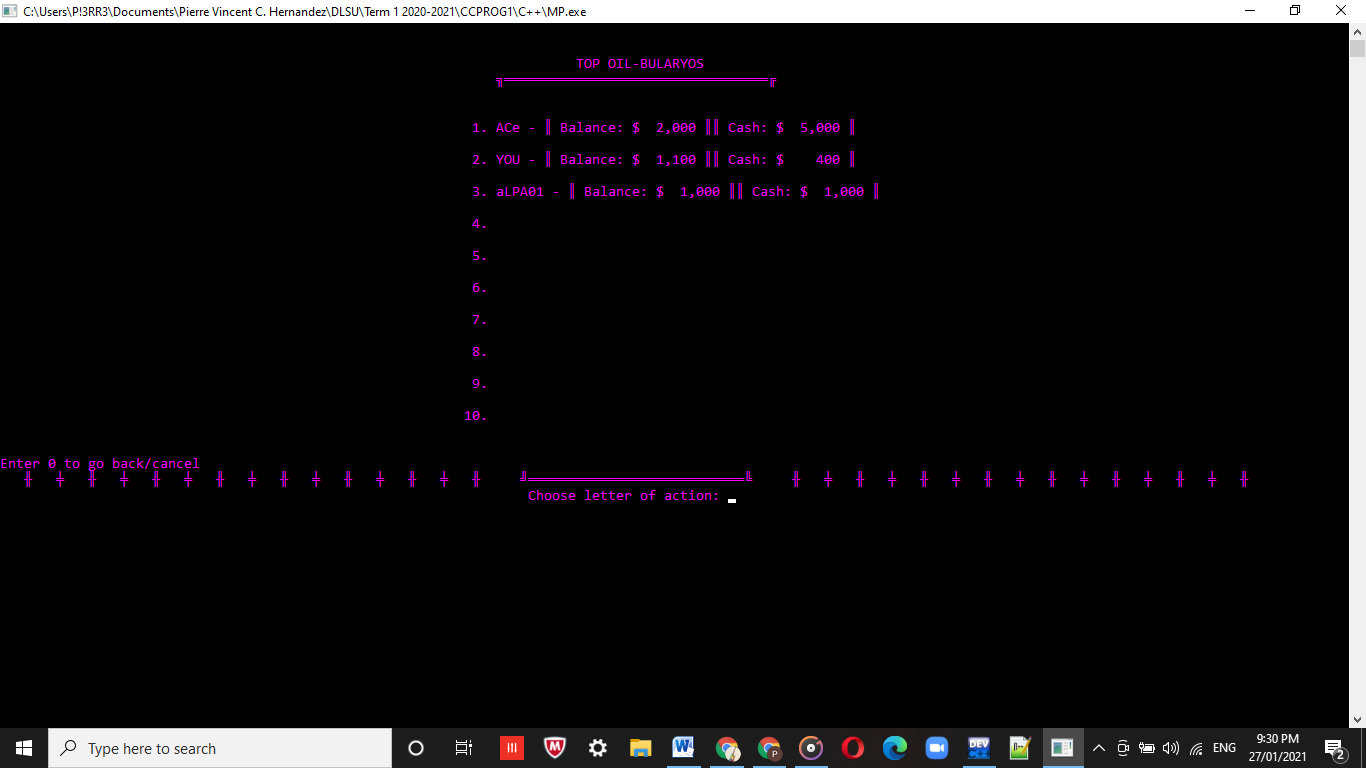
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FUNCTION:** void displayEssOilSellPrice(int nPriceOfProduct)  **DESCRIPTION:** displays the price(nPriceOfProduct) of essential and diffuser oils per drop | | | | | |
|  | **TEST DESCRIPTION** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| 1 | nPriceOfProduct is less than 1000 | nPriceOfProduct = 123 | 123/ drp | S 123/ drp | P |
| 2 | nPriceOfProduct is greater than or equal to 1000, and displays amount with comma | nPriceOfProduct = 2345 | 2,345/ drp | S 2,345/ drp | P |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FUNCTION:** void displayEssOilBuyProducts(char cOilIcon, const char \*ccOilName, int nCityStockPrice, int nCityDropPrice, int nCityStock, int nMaximumPurchBottle, char cLetter)  **DESCRIPTION:** displays the available Essential oils that can be bought base on the bottle stock left | | | | | |
|  | **TEST DESCRIPTION** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| 1 | nCityStock is 0; no stock left | nCityStock = 0 | NO DISPLAY | NO DISPLAY | P |
| 2 | NCityStock is greater than 0 | NCityStock = 5  *//Mint*  cOilIcon = MO  ccOilName = “Mint”  nCityStockPrice = 100  nCityDropPrice = 10  nMaximumPurchBottle = 5  cLetter = ‘M’ | M. Mint S 100 (S10/drp)[btl/s:5][10drps/btl] [max buy: 5] | M. Mint S 100 (S10/drp)[btl/s:5][10drps/btl] [max buy: 5] | P |

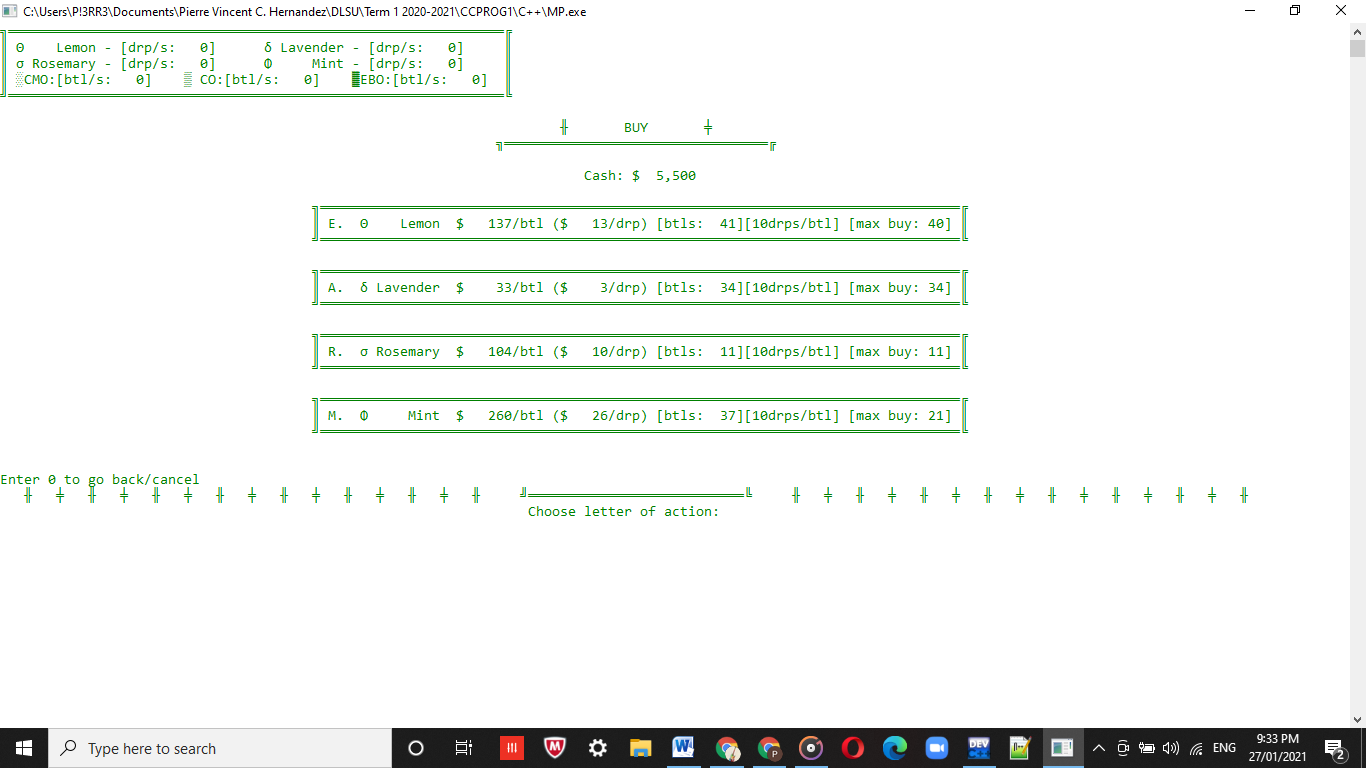
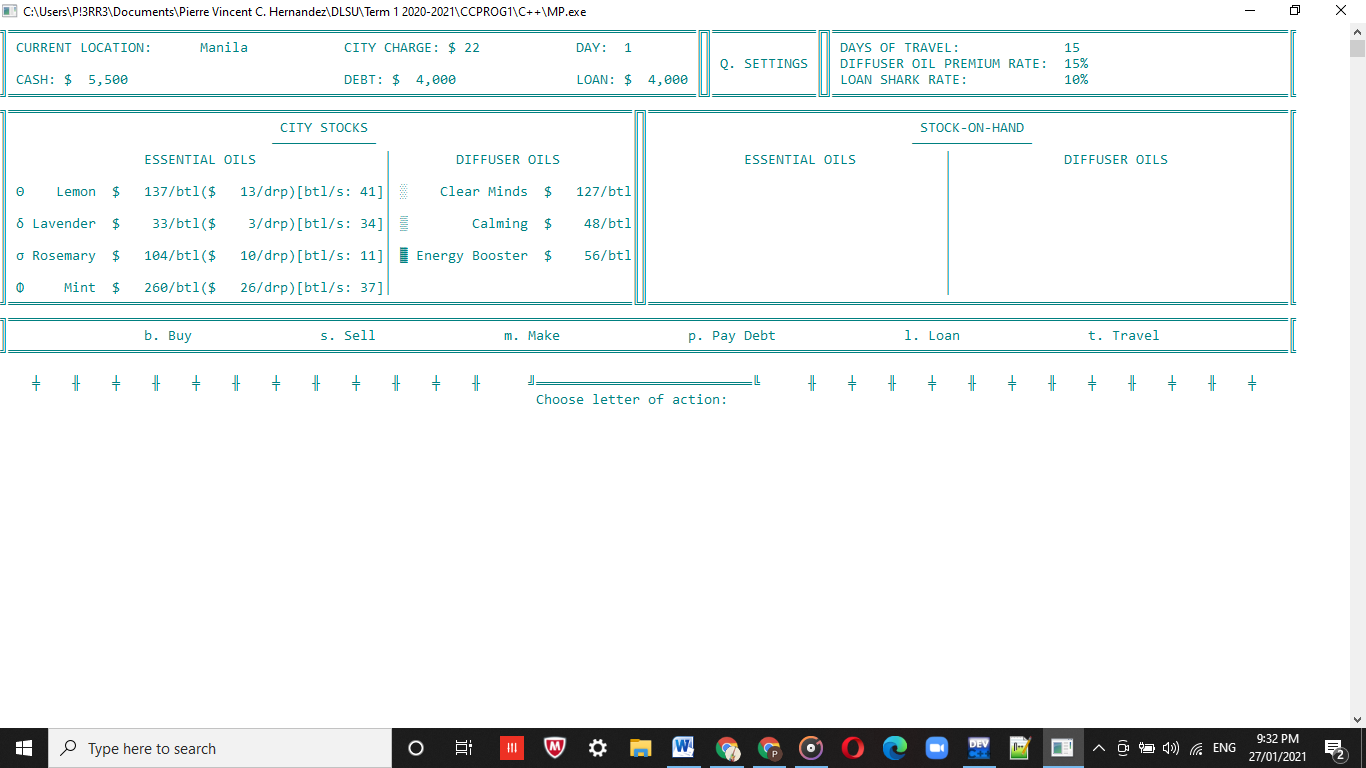
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FUNCTION:** void displayOilSellProducts(char cOilIcon, const char \*ccOilName, int nOilStockOnHand, int nOilSellPrice, char cOilLetter, char bOilType)  **DESCRIPTION:** displays the available Essential/Diffuser oils that can be sold base on the stock-on-hand drops/bottle left | | | | | |
|  | **TEST DESCRIPTION** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| 1 | nCityStock is 0; no stock left; bOilType is essential oil | nCityStock = 0  bOilType = ‘M’ (essential oil) | NO DISPLAY | NO DISPLAY | P |
| 2 | nCityStock is greater than 0; bOilType is essential oil | *//Mint*  cOilIcon = MO  nOilStockOnHand = 10  ccOilName = “Mint”  nOilSellPrice = 10  cOilLetter = ‘m’  bOilType = ‘M’ (essential oil) | m. Mint S 10/drp [stock: 10] [max sell: 10] | m. Mint S 10/drp [stock: 10] [max sell: 10] | P |
| 3 | nCityStock is 0; no stock left; bOilType is diffuser oil | nCityStock = 0  bOilType = ‘N’ (diffuser oil) | NO DISPLAY | NO DISPLAY | P |
| 4 | nCityStock is greater than 0; bOilType is diffuser oil | *//Calming*  cOilIcon = C  nOilStockOnHand = 5  ccOilName = “Calming”  nOilSellPrice = 45  cOilLetter = ‘y’  bOilType = ‘N’ (diffuser oil) | y. Calming S 45/btl [stock: 5] [max sell: 5] | y. Calming S 45/btl [stock: 5] [max sell: 5] | P |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FUNCTION:** void displayAvailableDiffOilMake(  const char \*ccDiffOilName, char cDiffOilIcon, int nDiffOilMakePrice, int nDiffOilMaxMake,  char cOilIngredientIcon1, char cOilIngredientIcon2, char cOilIngredientIcon3,  int nIngredientCount1, int nIngredientCount2, int nIngredientCount3,  char cDiffOilMakeLetter)  **DESCRIPTION:** displays the available Essential oils that can be bought base on the bottle stock left | | | | | |
|  | **TEST DESCRIPTION** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| 1 | nDiffOilMaxMake is 0; no diffuser oil of choice can be made | nDiffOilMaxMake = 0 | NO DISPLAY | NO DISPLAY | P |
| 2 | nDiffOilMaxMake is greater than 0; diffuser oil can be made | ccDiffOilName = “Calming”  *//Calming*  cDiffOilIcon = C  nDiffOilMakePrice = 35  nDiffOilMaxMake = 3  *//Lavender*  cOilIngredientIcon1 = LAO  *//Mint*  cOilIngredientIcon2 = MO  *//Lemon*  cOilIngredientIcon3 = LEO  nIngredientCount1 = 3  nIngredientCount2 = 2  nIngredientCount3 = 1  cDiffOilMakeLetter = ‘Y’ | M. Calming S 35 [LAO/MO/LEO: 3/2/1] [max make: 3] | M. Calming S 35 [LAO/MO/LEO: 3/2/1] [max make: 3] | P |

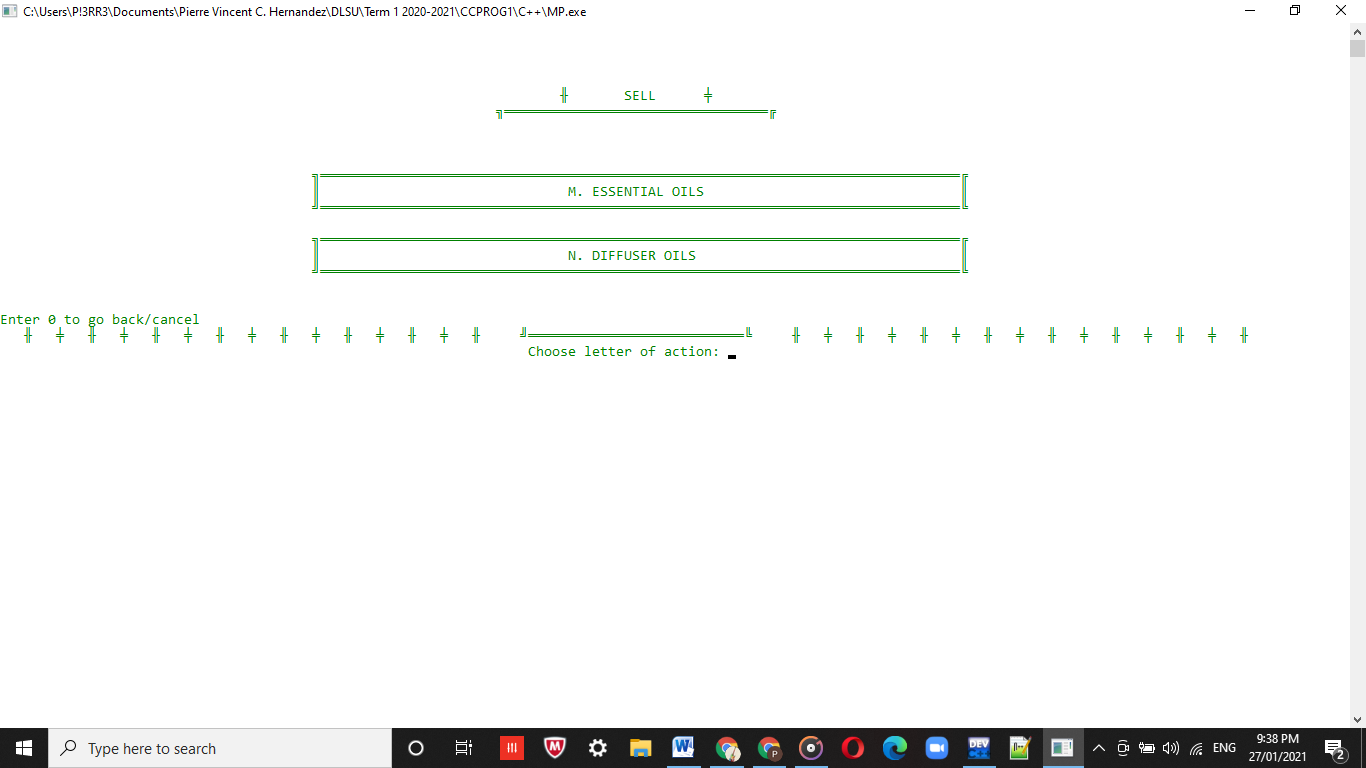
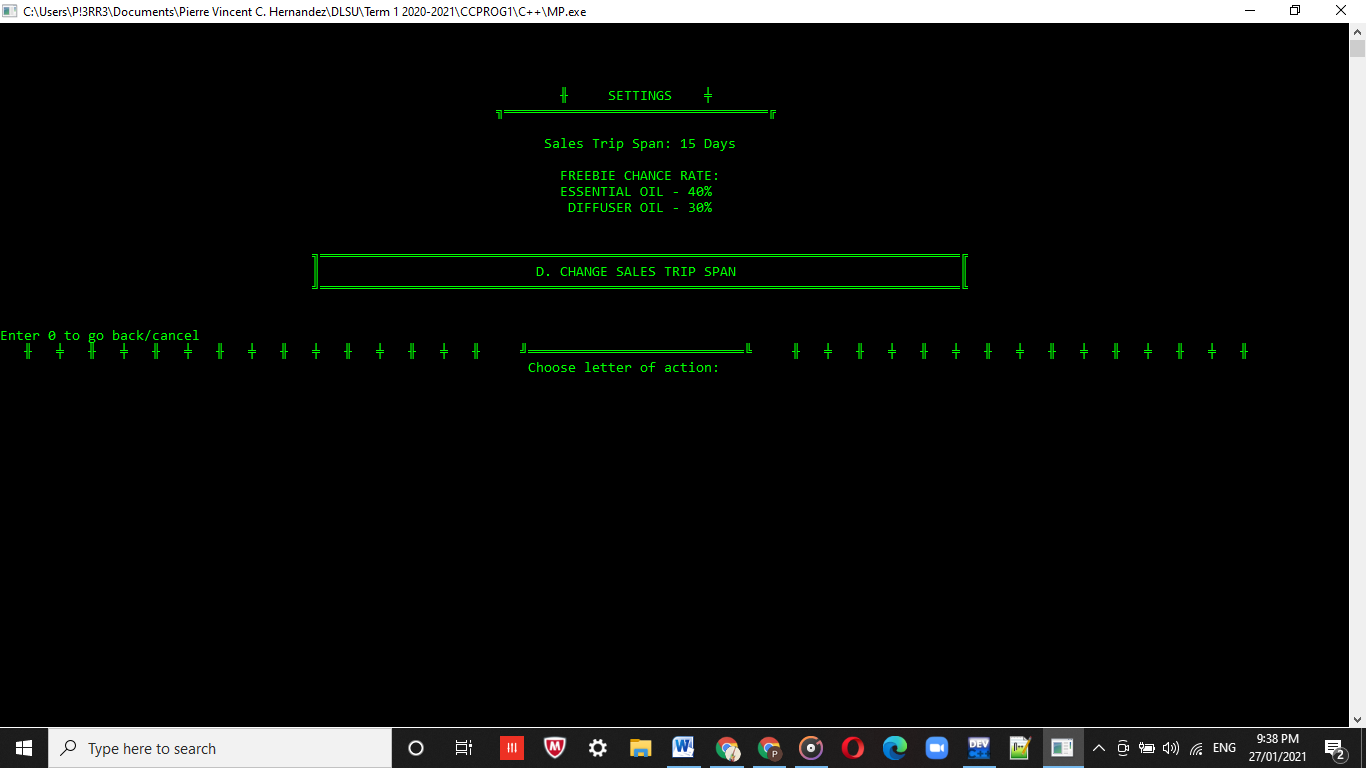
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FUNCTION:** void displayTravelChoices(int bPlace)  **DESCRIPTION:** displays the locations(except the player’s current location) where the player could travel | | | | | |
|  | **TEST DESCRIPTION** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| 1 | Player & bPlace is in Manila | bPlace = 1 | G. MAKATI  H. ALABANG  I. QUEZON CITY | G. MAKATI  H. ALABANG  I. QUEZON CITY | P |
| 2 | Player & bPlace is in Makati | bPlace = 2 | F. MANILA  H. ALABANG  I. QUEZON CITY | F. MANILA  H. ALABANG  I. QUEZON CITY | P |
| 3 | Player & bPlace is in Alabang | bPlace = 3 | F. MANILA  G. MAKATI  I. QUEZON CITY | F. MANILA  G. MAKATI  I. QUEZON CITY | P |
| 4 | Player & bPlace is in Quezon City | bPlace = 4 | F. MANILA  G. MAKATI  H. ALABANG | F. MANILA  G. MAKATI  H. ALABANG | P |

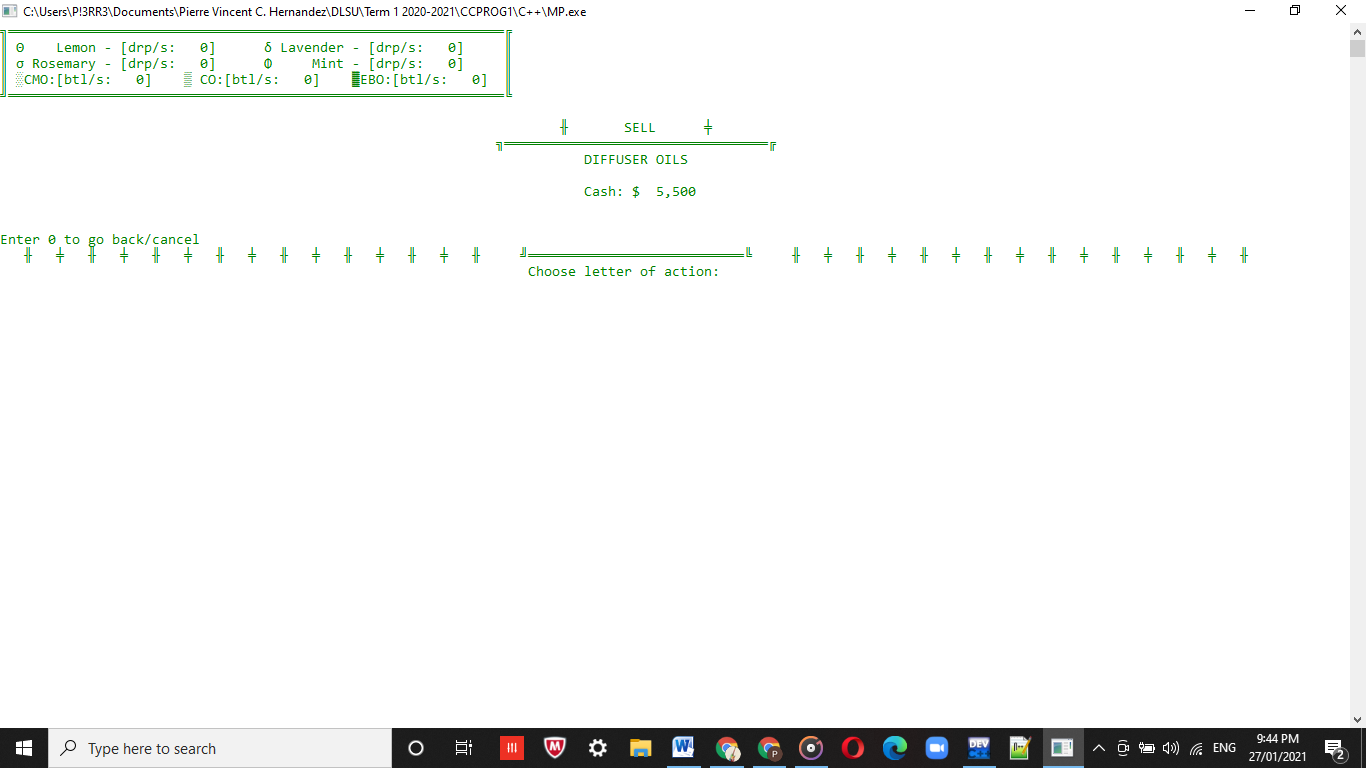
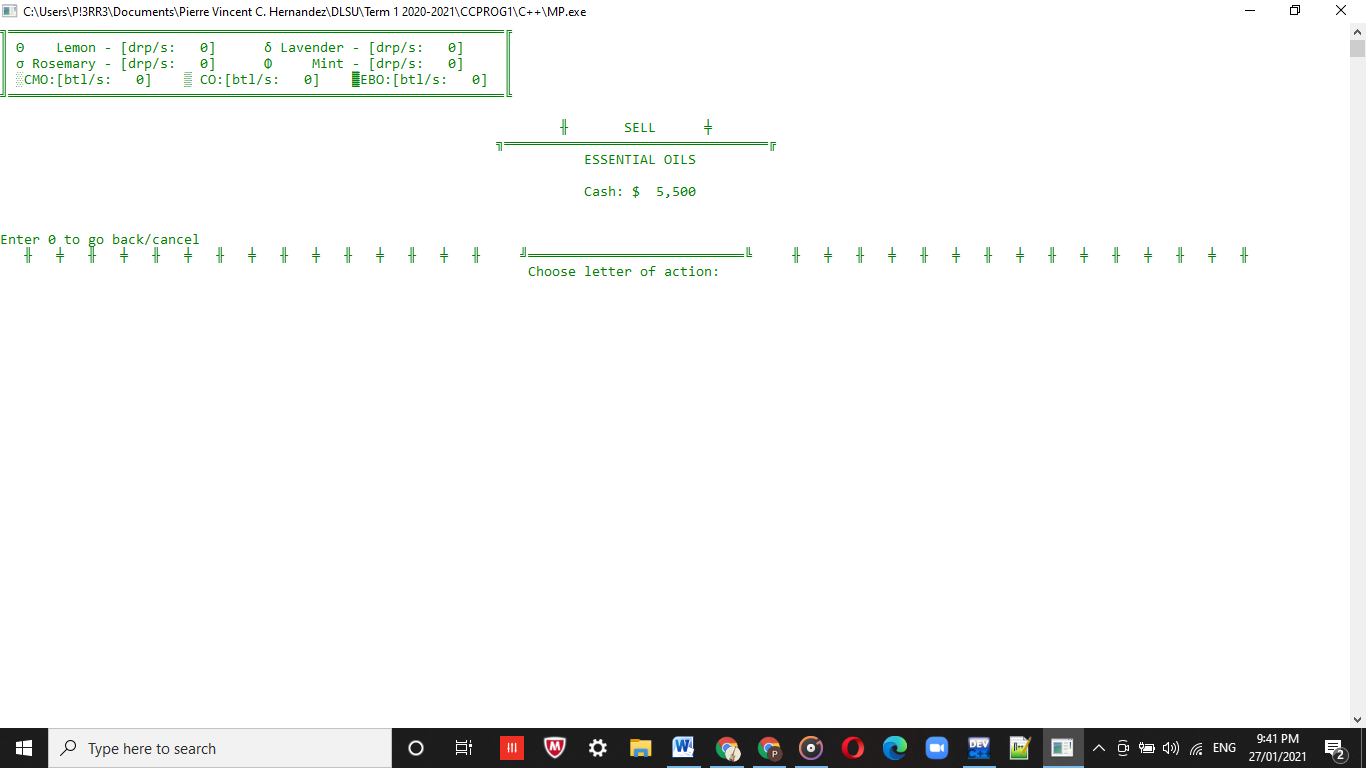
**void executeOilbularyoGame() void displayTopTenList(/\*see code for parameters\*/)**



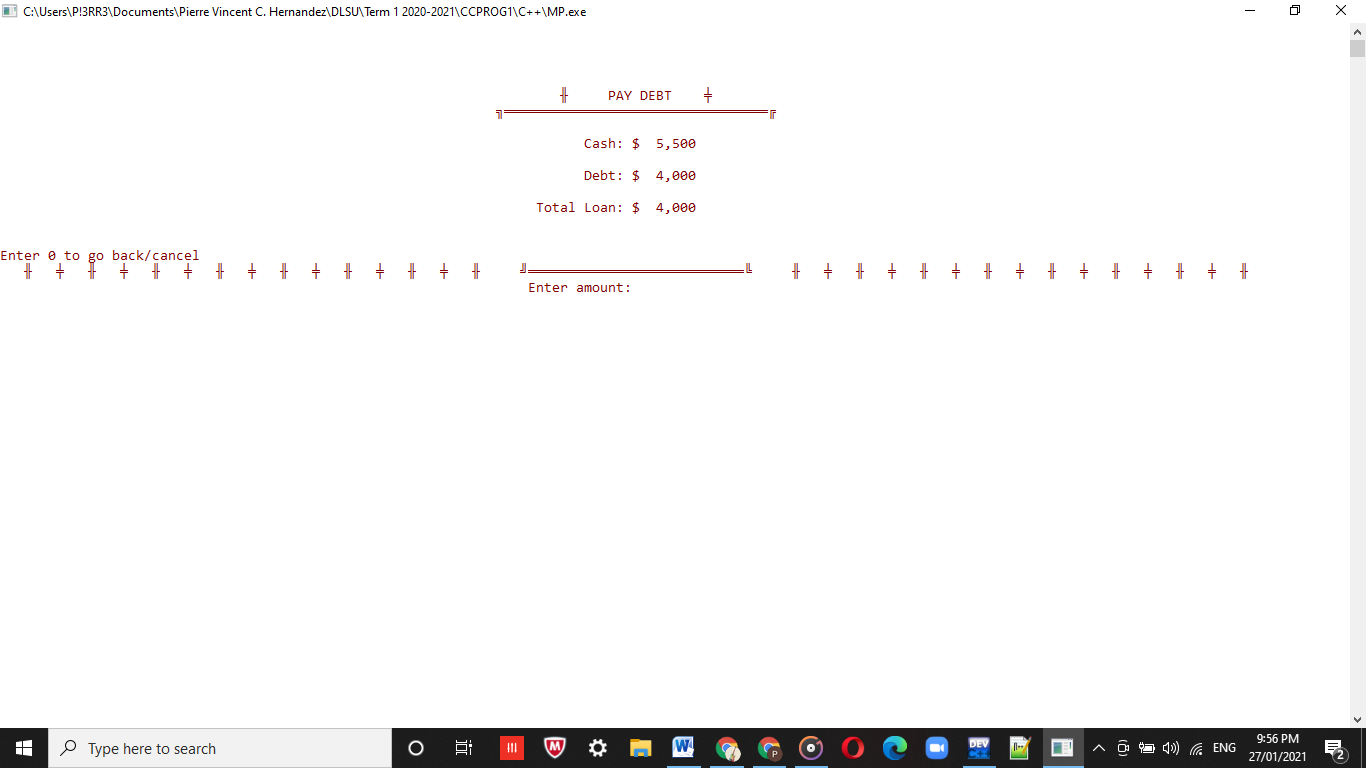
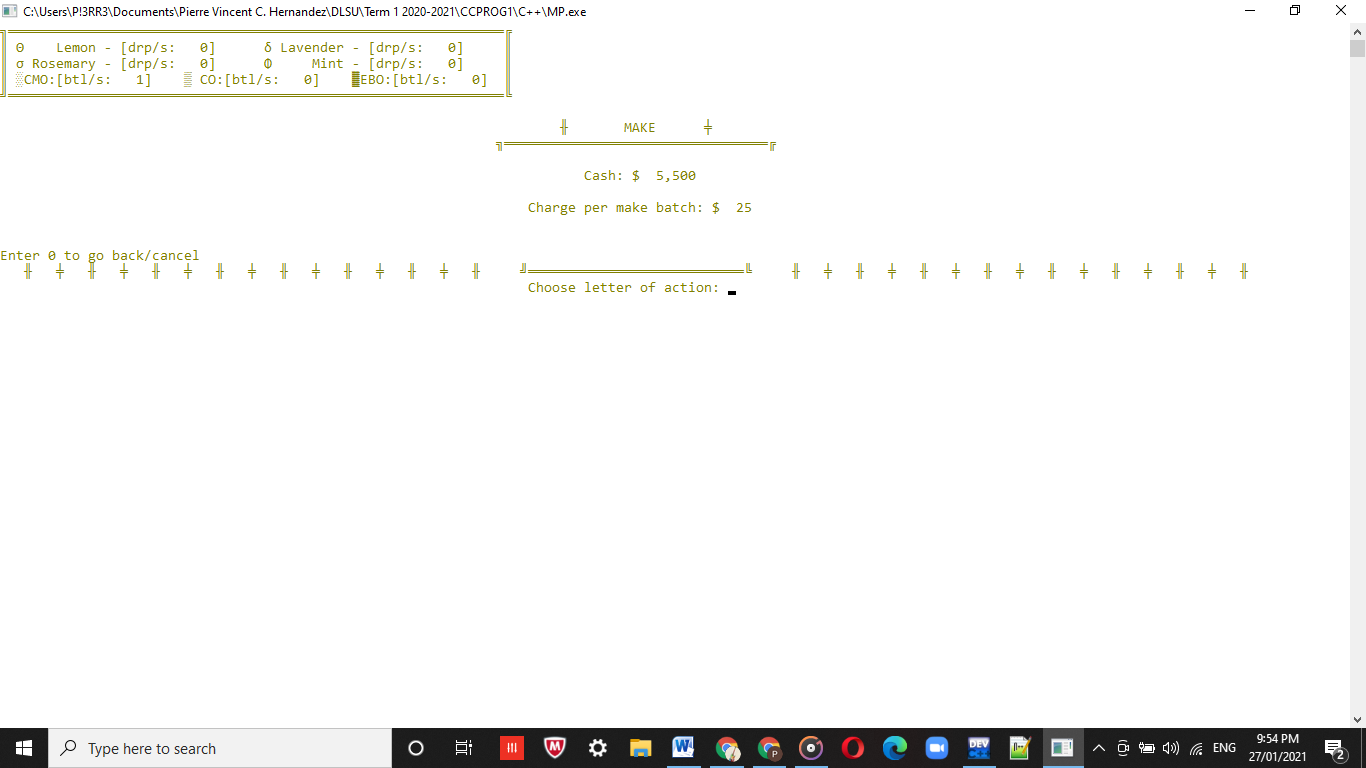
**void displayGameMenu(/\*see code for parameters\*/) void displayBuyMenu(/\*see code for parameters\*/)**



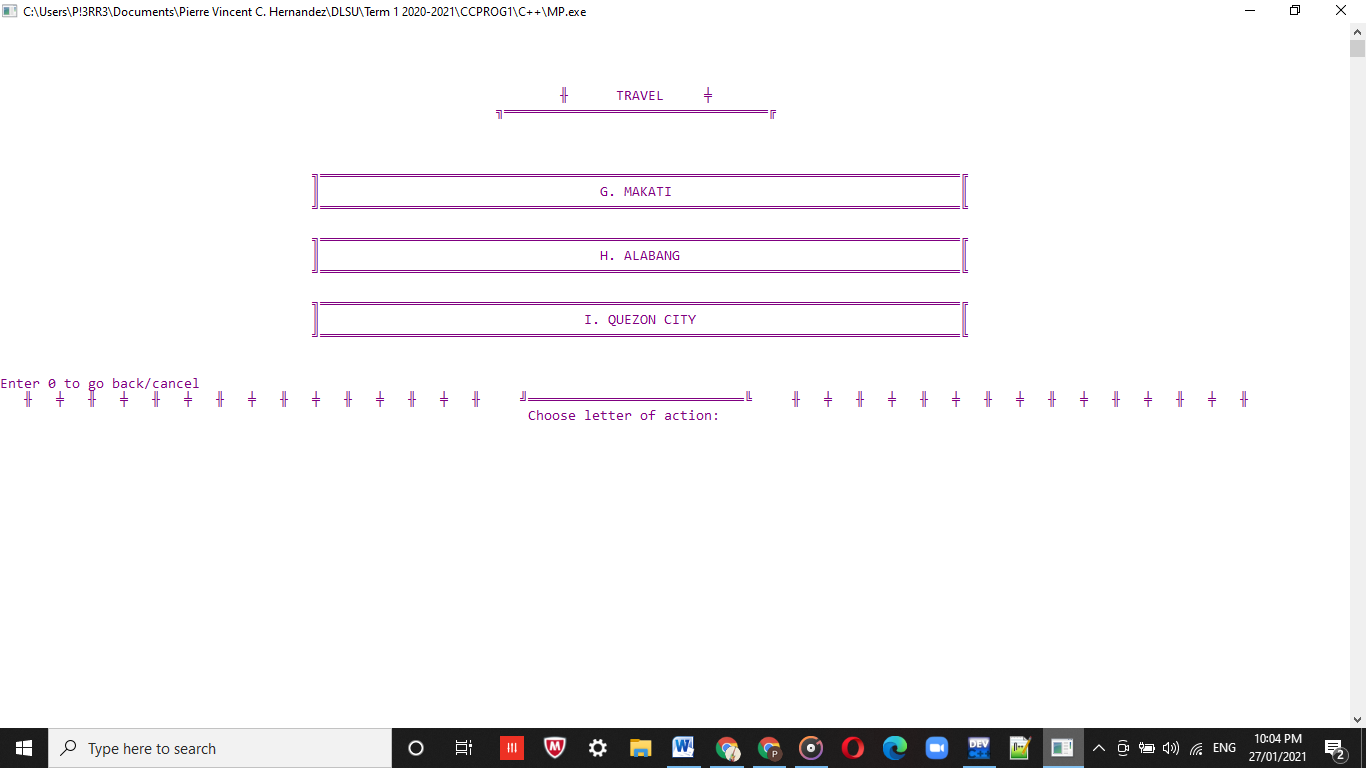
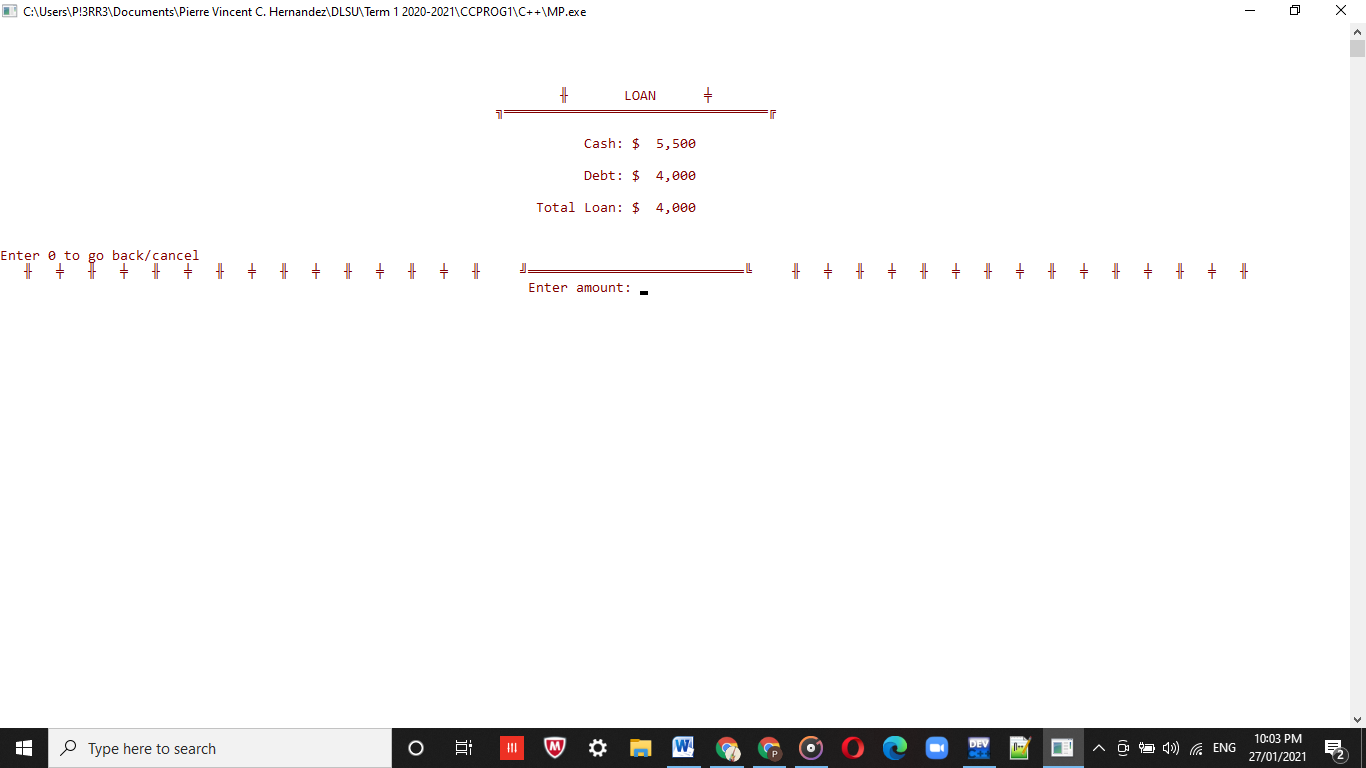
**void displaySettingsMenu(/\*see code for parameters\*/) void displaySellMainMenu(/\*see code for parameters\*/)**



**void displaySellEssOilMenu(/\*see code for parameters\*/) void displaySellDiffOilMenu(/\*see code for parameters\*/)**



**void displayMakeMenu(/\*see code for parameters\*/) void displayPayDebtMenu(/\*see code for parameters\*/)**



**void displayLoanMenu(/\*see code for parameters\*/) void displayTravelMenu(/\*see code for parameters\*/)**

