Source Code: <https://github.com/UAMIS221-321/mis221-pa5-Dnsavage>

Additional Features

* Unexpected data type handling
  + Gives an error and repeats instructions if unexpected data type is received
    - E.g.: string, character, or null instead of integer
* Unexpected input handling
  + Gives an error and repeats instructions if unexpected value entered
    - E.g.: negative or zero when entering a menu selection
* Deletion Checking
  + User is asked if they are sure when attempting to delete items
* Format Headers
  + Displays with Multiple Fields have Format Headers so user can see which field is what
* Clean Slate
  + Deleting the current vacation clears all previously associated activity data to prepare for the next vacation.
    - Warns user that all associated data with the vacation will also be deleted
    - Report files remain intact until a new report is successfully created and saved to the file.
    - Clears list.txt
    - Clears completed.txt
* No activities without vacation
  + Cannot add activities if no vacation is defined
* Vacation Destination Filling
  + If no destination is entered, a place holder [No Destination] is provided
* Activity Completion Review Filling
  + If no completion review is entered, a place holder [No Review] is provided
* Remaining Activities Extended Sort
  + Required sorting methods can be in ascending or descending order
* Manual Date Entry (MM/DD/YYYY)
  + Vacation start and end dates and activity completion dates are manually entered.
  + Vacation end dates are checked against vacation start dates
  + Activity completion dates must be within or on the vacation start and end dates
  + All dates are checked against the number of days in the selected month
  + Dates cannot come before 1999 or after 2099
  + All dates subject to null/unexpected error handling
* Manual Activity Time Entry ([integer]-minutes/hours/days)
  + User is prompted for a completed activity’s length via a series of menus
  + First asked whether it takes minutes, hours, or days to complete
  + Then asked how many of the selected time format
  + All activity lengths are converted to minutes for streamlined sorting
  + All times are subject to null/unexpected error handling

Additional Features (Continued)

* Multilanguage Compatibility (English, Spanish, French, German)
  + All non-user-created displays are translatable to English, Spanish, French, and German
  + Language chosen at beginning of program via language menu
  + Language may be changed at any time by returning to Main Menu and selecting “Change Language”
  + Once a language is chosen, a static language ID is assigned
  + Multilanguage Compatibility depends on a complicated menu system
    - All menus are assigned a Prompt Value that determines the number of lines of prompts for a given menu
    - All menus are assigned an Options Value that determines the number of options for a given menu
    - All menus are assigned a Menu Type that determines which line to find a given file path on
    - For any given menu, the display is retrieved via the following process:
      1. A file containing each menu file path is opened
      2. The file line count correlates with the Menu Type
      3. The line containing the selected menu file path is sent to processing where it is opened
      4. Each line of the opened file is split by a delimiter, the resulting elements of which follow:
      5. Element 1: Language ID; Compared with static language ID to determine which lines are appropriate for the user’s selected language
      6. Element 2: Line type; used to determine whether a given line is a prompt or an option
      7. Element 3: Display; What the user actually sees
      8. Appropriate Prompts are stored under the “displays” object name
      9. Appropriate Options are stored under the “menuOptions” object name
    - For any given menu, the display is achieved via the following process:
      1. Display each prompt according to the Prompts Count
      2. Display each option according to the menuOptions Count
    - For any given menu, user selection is achieved via the following process:
      1. User’s option selection, the number of options, and the actual options are all sent to an error handler
      2. If user selection is null or not listed, an error message is shown, followed by the actual options, which are stored in an array and then iterated through according to the number of options to be displayed back to the user

Additional Features (Continued)

* + Category queries are handled via the file-handling system:
    - 1. When a category is requested by the system, the file containing all categories in all compatible languages is opened, iterated through with the language ID as the search value, and stopped at the first instance of a match
      2. Once a language match is reached, the system iterates through the next X lines, where X is the number of categories
      3. If the given category index matches the one in the current iteration, the field is split, and the category name returned to the system in the appropriate language
  + All non-menu standard text displays are stored in methods containing switch statements in a Prompts class, the switch variables of which is the static language ID
    - When output is needed on the console, the needed Prompt is called, and the static language ID determines the appropriate language to display the Prompt in
* ASCII Art messages
  + Displays an ASCII “Happy travels!” in the user’s selected language upon program exit
  + Displays a random ASCII image upon program exit. Images include:
    - Stonehenge
    - Eiffel Tower
    - Taj Mahal
    - Statue of Liberty
    - Space Needle
  + ASCII images are stored in .txt files and iterated through when chosen
* Text File Verification System
  + At the beginning of the program, ensure that all necessary .txt assets are in place.
    - Checks that all needed .txt files exist
    - Prevents execution of rest of program in case a needed file is missing
    - If a file is missing, prompts the user to restore the specified file(s)
    - Displays error in English, Spanish, French, and German
  + Useful in case user deletes/moves a needed asset
  + Checks the names of current files against those held in a .txt file hosted on GitHub

|  |  |  |
| --- | --- | --- |
| **DisplayMainMenu** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Instantiate mainMenuDisplays as a Prompts[], length 2  Instantiate mainMenuOptions as a MenuDisplays[], length 7  Call SetMenuType from MenuDisplays class, pass 0  Instantiate mainMenu as a DisplaysFile, pass mainMenuDisplays and  mainMenuOptions  Call GetMenuDisplay from mainMenu  Instantiate displayCurrent as a DisplaysReport, pass mainMenuDisplays and  mainMenuOptions  Call DisplayText from displayCurrent  Set menuChoice as an int = GetMainMenuChoice, pass mainMenuOptions  Call RouteMainMenu, pass menuChoice | Main Menu |

|  |  |  |
| --- | --- | --- |
| **GetMainMenuChoice** | | |
| **Input** | **Process** | **Output** |
| mainMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass  mainMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate mainMenu as a Menu  Set mainMenu = a new Menu  Call SetNumOptions of mainMenu, pass length of  mainMenuOptions  Call SetOptions of mainMenu, pass options  Set menuChoice = GetValidMenuChoice of mainMenu  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **RouteMainMenu** | | |
| **Input** | **Process** | **Output** |
| menuChoice | Processing Items:  None  Algorithm:  Declare a switch statement, variable menuChoice  in case 1: Call DisplayVacationMenu  in case 2: Call DisplayActivityMenu  in case 3: Call DisplayRemainingMenu  in case 4: Call DisplayCompleteMenu  in case 5: Call DisplayTripReportMenu  in case 6: Call DisplayLanguageMenu  default to: Call ThankUser of Prompts  Call DisplayExitMessage of BigPicture  Call DisplayExitImage of BigPicture  Exit the program  End switch | Vacation Menu  Activity Menu  Remaining Menu  Complete Menu  Report Menu  Language Menu  Program exit message |

|  |  |  |
| --- | --- | --- |
| **DisplayLanguageMenu** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Call LanguageMenu of Prompts  Set menuChoice as an int = GetLangaugeChoice  Call SetLanguageID of Prompts, pass menuChoice  Clear the console  Call DisplayMainMenu | Language Menu |

|  |  |  |
| --- | --- | --- |
| **GetLanguageChoice** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Set options as a string[] = LanguageOptions of Prompts  Instantiate launguageMenu as a Menu  Set languageMenu = a new Menu  Call SetNumOptions of languageMenu, pass length of options  Call SetOptions of languageMenu, pass options  Set menuChoice as an int = GetValidMenuChoice of languageMenu  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **DisplayVacationMenu** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Clear the console  Instantiate vacations as a Vacation, length 3  Instantiate vacationMenuDisplays as a Prompts[], length 1  Instantiate vacationMenuOptions as a MenuDisplays[], length 3  Call SetMenuType of MenuDisplays, pass 1  Instantiate vacationMenu as a DisplaysFile, pass vacationMenuDisplays  vacationMenuOptions  Call GetMenuDisplay of vacationMenu  Instantiate displayCurrent as a DisplaysReports, pass vacationMenuDisplays  , vacationMenuOptions  Call DisplayText of displayCurrent  Set vacMenuChoice as an int = GetVacmenuChoice, pass  vacationMenuOptions  Call RouteVacMenuChoice, pass vacMenuChoice and vacations | Vacation Menu |

|  |  |  |
| --- | --- | --- |
| **GetVacMenuChoice** | | |
| **Input** | **Process** | **Output** |
| vacationMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass  vacationMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate vacationMenu as a Menu  Set vacationMenu = a new Menu  Call SetNumOptions of vacationMenu, pass length of  vacationMenuOptions  Call SetOptions of vacationMenu, pass options  Set menuChoice as an int = GetValidMenuChoice of  vacationMenu  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **RouteVacMenuChoice** | | |
| **Input** | **Process** | **Output** |
| vacMenuChoice  vacations | Processing Items:  None  Algorithm:  Declare a switch statement, variable vacMenuChoice  in case 1: Instantiate newVacation as a VacationFile, pass  vacations  Call GetAllVacations of newVacation  If GetCount of Vacation < 1  Call getNewVacationInfo of newVacation, pass  Vacations  Else  Call PromptCantAddVacation of Prompts  End if  Call DisplayVacationMenu  in case 2: Call DisplayEditVacMenu, pass vacations  default to: Clear the console  Call DisplayMainMenu  End switch | New Vacation prompt  Edit Vacation Menu  Main Menu |

|  |  |  |
| --- | --- | --- |
| **DisplayEditVacMenu** | | |
| **Input** | **Process** | **Output** |
| vacations  vacEditID | Processing Items:  None  Algorithm:  Clear the console  Instantiate editVacMenu as a VacationFile, pass vacations  Call GetAllVacations of editVacMenu  Instantiate checkForVacations as a VacationReport  If CheckIfAnyVacations, passed vacations, of checkForVacations < 0  Call DisplayVacationMenu  End if  Call PromptForVacID of Prompts  Input vacEditID  If vacEditID in lower-case == “stop”  Call DisplayVacationMenu  End if  Set lineToEdit as an int = GetUserLineToEdit of editVacMenu, pass  vacEditID  if lineToEdit < 0  Call DisplayAbortEditMessage of Prompts  Call DisplayVacationMenu  End if  Call EditingMessage of Prompts  Instantiate singleLine as a VacationReport, pass vacations  Call DisplayOneVacation of singleLine, pass lineToEdit  Call VacEditOrDeleteMenu, pass lineToEdit, vacations, and singleLine | Edit Vacation Menu |

|  |  |  |
| --- | --- | --- |
| **VacEditOrDeleteMenu** | | |
| **Input** | **Process** | **Output** |
| lineToEdit  vacations  singleLine | Processing Items:  None  Algorithm:  Instantiate vacEditOrDeleteMenuDisplays as a Prompts[], length 1  Instantiate vacEditOrDeleteMenuOptions as a MenuDisplays[], length 3  Call SetMenuType of MenuDisplays, pass 2  Instantiate vacationMenu as a DisplaysFile, pass  vacEditOrDeleteMenuDisplays and vacEditOrDeleteMenuOptions  Call GetMenuDisplay of vacationMenu  Instantiate displayCurrent as a DisplaysReports, pass  vacEditOrDeleteMenuDisplays and vacEditOrDeleteMenuOptions  Call DisplayText of displayCurrent  Set editOrDeleteChoice as an int = GetEditOrDeleteChoice, pass  vacEditOrDeleteMenuOptions  Call RouteEditOrDelete, pass editOrDeleteChoice, lineToEdit, vacations  , and singleLine | Vacation Edit/Delete Menu |

|  |  |  |
| --- | --- | --- |
| **GetEditOrDeleteChoice** | | |
| **Input** | **Process** | **Output** |
| vacEditOrDeleteMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass vacEditOrDeleteMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate editOrDelete as a Menu  Set editOrDelete = a new Menu  Call SetNumOptions of editOrDelete, pass length of  vacEditOrDeleteMenuOptions  Call SetOptions of editOrDelete, pass options  Set menuChoice as an int = GetValidMenuChoice of  editOrDelete  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **RouteEditOrDelete** | | |
| **Input** | **Process** | **Output** |
| editOrDeleteChoice  lineToEdit  vacations  singleLine | Processing Items:  None  Algorithm:  Declare a switch statement, variable editOrDeleteChoice  in case 1: Call DisplayEditFieldMenu, pass lineToEdit,  vacations, and singleLine  in case 2: Call DeleteVacationChoice, pass lineToEdit,  vacations, and singleLine  default to: Call DisplayVacationMenu  End switch | Edit Field Menu  Delete Vacation  Choice Menu  Vacation Menu |

|  |  |  |
| --- | --- | --- |
| **DisplayEditFieldMenu** | | |
| **Input** | **Process** | **Output** |
| lineToEdit  vacations  singleLine | Processing Items:  None  Algorithm:  Call EditingMessage of Prompts  Call DisplayOneVacation of singleLine, pass lineToEdit  Instantiate editFieldMenuDisplays as a Prompts[], length 1  Instantiate editFieldMenuOptions as a MenuDisplays[], length 5  Call SetMenuType of MenuDisplays, pass 3  Instantiate vacationMenu as a DisplaysFile, pass  editFieldMenuDisplays and editFieldMenuOptions  Call GetMenuDisplay of vacationMenu  Instantiate displayCurrent as a DisplaysReports, pass  editFieldMenuDisplays and editFieldMenuOptions  Call DisplayText of displayCurrent  Set fieldEditChoice as an int = GetEditOrDeleteChoice, pass  vacEditOrDeleteMenuOptions  Call RouteEditOrDelete, pass lineToEdit, fieldEditChoice, vacations | Edit Field Menu |

|  |  |  |
| --- | --- | --- |
| **GetEditVacFieldChoice** | | |
| **Input** | **Process** | **Output** |
| editFieldMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass  editFieldMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate editVacationField as a Menu  Set editVacationField = a new Menu  Call SetNumOptions of editVacationField, pass length of  vacEditOrDeleteMenuOptions  Call SetOptions of editVacationField, pass options  Set menuChoice as an int = GetValidMenuChoice of  editVacationField  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **EditVacationField** | | |
| **Input** | **Process** | **Output** |
| lineToEdit  fieldEditChoice  vacations | Processing Items:  None  Algorithm:  If fieldEditChoice = 6  Call DisplayMainMenu  Else  Instantiate newEdit as a VacationUtil, pass vacations  Call EditVacationField of newEdit, pass fieldEditChoice  and lineToEdit  Call EditVacationToFile of VacationFile, pass vacations  Call DisplayVacationMenu  End if | Updated  Vacation field  Main Menu |

|  |  |  |
| --- | --- | --- |
| **DeleteVacationChoice** | | |
| **Input** | **Process** | **Output** |
| lineToEdit  vacations  singleLine | Processing Items:  None  Algorithm:  Call DeletingMessage of Prompts  Call DisplayOneVacation of singleLine, pass lineToEdit  Instantiate deleteVacMenuDisplays as a Prompts[], length 3  Instantiate deleteVacMenuOptions as a MenuDisplays[], length 2  Call SetMenuType of MenuDisplays, pass 4  Instantiate vacationMenu as a DisplaysFile, pass  deleteVacMenuDisplays and deleteVacMenuOptions  Call GetMenuDisplay of vacationMenu  Instantiate displayCurrent as a DisplaysReports, pass  deleteVacMenuDisplays and deleteVacMenuOptions  Call DisplayText of displayCurrent  Set deleteVerify as an int = GetDeleteVerify, pass  deleteVacMenuOptions  Call RouteDeleteChoice, pass deleteVerify, lineToEdit, vacations | Delete Vacation  Menu |

|  |  |  |
| --- | --- | --- |
| **GetDeleteVerify** | | |
| **Input** | **Process** | **Output** |
| deleteVacMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass  deleteVacMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate verifyDeleteVacation as a Menu  Set verifyDeleteVacation = a new Menu  Call SetNumOptions of verifyDeleteVacation, pass length of  deleteVacMenuOptions  Call SetOptions of verifyDeleteVacation, pass options  Set menuChoice as an int = GetValidMenuChoice of  verifyDeleteVacation  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **RouteDeleteChoice** | | |
| **Input** | **Process** | **Output** |
| deleteVerify  lineToEdit  vacations | Processing Items:  None  Algorithm:  If deleteVerify = 1  Call DeleteAllCompletionsFromFile of CompleteActivityFile  Call DeleteAllActivitiesFromFile of ActivityFile  Call DeleteVacationFromFile of VacationFile, pass lineToEdit,  vacations  Call deleteVacMessage of Prompts  Call DisplayVacationMenu  Else  Call AbortVacDeletionMessage of Prompts  Call DisplayVacationMenu  End if | Deleted  Vacation  Deleted  Activities Info  Vacation Menu |

|  |  |  |
| --- | --- | --- |
| **DisplayActivityMenu** | | |
| **Input** | **Process** | **Output** |
| none | Processing Items:  None  Algorithm:  Clear the console  Instantiate activityMenuDisplays as a Prompts[], length 1  Instantiate activityMenuOptions as a MenuDisplays[], length 4  Call SetMenuType of MenuDisplays, pass 5  Instantiate activityMenu as a DisplaysFile, pass  activityMenuDisplays and activityMenuOptions  Call GetMenuDisplay of activityMenu  Instantiate displayCurrent as a DisplaysReports, pass  activityMenuDisplays and activityMenuOptions  Call DisplayText of displayCurrent  Set actMenuChoice as an int = GetActMenuChoice, pass  activityMenuOptions  Call RouteActMenuChoice, pass actMenuChoice | Activity Menu |

|  |  |  |
| --- | --- | --- |
| **GetActMenuChoice** | | |
| **Input** | **Process** | **Output** |
| activityMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass  deleteVacMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate activityMenu as a Menu  Set activityMenu = a new Menu  Call SetNumOptions of activityMenu, pass length of  deleteVacMenuOptions  Call SetOptions of activityMenu, pass options  Set menuChoice as an int = GetValidMenuChoice of  activityMenu  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **RouteActMenuChoice** | | |
| **Input** | **Process** | **Output** |
| actMenuChoice | Processing Items:  None  Algorithm:  Instantiate activities as a new Activity[], length 50  Declare a switch statement, variable actMenuChoice  In case 1: Call ViewActivities, pass activities  In case 2: Call NewActivity, pass activities  In case 3: Call DisplayEditActMenu, pass activities  Default to: Clear the console  Call DisplayMainMenu  End switch | View Activities  Display  New Activity  Prompt  Edit Activity  Menu  Main Menu |

|  |  |  |
| --- | --- | --- |
| **ViewActivities** | | |
| **Input** | **Process** | **Output** |
| activities | Processing Items:  None  Algorithm:  Clear the console  Instantiate activityFill as an ActivityFile, pass activities  Call GetAllActivities of activityFill  Instantiate checkForVacations as an ActivityReport  If CheckIfAnyActivities, passed activities, of checkForVacations < 0  Call DisplayActivityMenu  End if  Call PromptReturnToActMenu of Prompts  Call DisplayActivityMenu | All activities  Activity Menu |

|  |  |  |
| --- | --- | --- |
| **NewActivity** | | |
| **Input** | **Process** | **Output** |
| activities  activityDetail | Processing Items:  None  Algorithm:  Insantiate indexVac as a Vacation[], length 3  Instantiate countVac as a VacationFile, pass indexVac  Call GetAllVacations of countVac  If GetCount of Vacation < 1  Call PromptNeedVacation of Prompts  Call DisplayMainMenu  End if  Call PromptActDetails of Prompts  Instantiate newID as an ActivityFile, pass activities  Call GetAllActiviites of newID  Set activities[GetCount of Activity] = new Activity  Call SetActID, passed GenActivityID of newID, of  activities[GetCount of Activity]  Call PromptName of Prompts  Input activityDetail  Call SetName, passed activityDetail, of  activities[GetCount of Activity]  Call DisplayCategoryMenu, pass activities  Call PromptMinPrice of Prompts  Instantiate newActivityCheck as an ActivityUtil  Set activityDetail = GetValidInteger of newActivityCheck  Call SetPriceMin, passed activityDetail, of  activities[GetCount of Activity]  Call PromptMaxPrice of Prompts  Set activityDetail = GetValidInteger of newActivityCheck  Set activityDetail = CheckMaxPrice, passed  activityDetail and activities, of newActivityCheck  Call SetPriceMax, passed activityDetails, of  activities[GetCount of Activity]  Call DisplayTimeMenu, pass activities  CallDisplayTicketMenu, pass activities  Call NewActivityToFile, passed activities, of ActivityFile  Call IncCount of Activity  Call DisplayActivityMenu | Prompt for new activity details |

|  |  |  |
| --- | --- | --- |
| **DisplayCategoryMenu** | | |
| **Input** | **Process** | **Output** |
| activities | Processing Items:  None  Algorithm:  Clear the console  Instantiate catMenuDisplays as a Prompts[], length 1  Instantiate catMenuOptions as a MenuDisplays[], length 9  Call SetMenuType of MenuDisplays, pass 6  Instantiate activityMenu as a DisplaysFile, pass  catMenuDisplays and catMenuOptions  Call GetMenuDisplay of activityMenu  Instantiate displayCurrent as a DisplaysReports, pass  catMenuDisplays and catMenuOptions  Call DisplayText of displayCurrent  Set catMenuChoice as an int = GetActMenuChoice, pass  catMenuOptions  Instantiate catAssign as an ActivityUtil  Call AssignCategory, passed catMenuChoice, activities, of catAssign | Activity  Category Menu |

|  |  |  |
| --- | --- | --- |
| **GetCatMenuChoice** | | |
| **Input** | **Process** | **Output** |
| catMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass  catMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate categoryMenu as a Menu  Set categoryMenu = a new Menu  Call SetNumOptions of categoryMenu, pass length of  catMenuOptions  Call SetOptions of categoryMenu, pass options  Set menuChoice as an int = GetValidMenuChoice of  categoryMenu  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **DisplayTimeMenu** | | |
| **Input** | **Process** | **Output** |
| activities | Processing Items:  None  Algorithm:  Clear the console  Instantiate timeMenuDisplays as a Prompts[], length 1  Instantiate timeMenuOptions as a MenuDisplays[], length 3  Call SetMenuType of MenuDisplays, pass 7  Instantiate getTimeMenu as a DisplaysFile, pass  timeMenuDisplays and timeMenuOptions  Call GetMenuDisplay of activityMenu  Instantiate displayCurrent as a DisplaysReports, pass  timeMenuDisplays and timeMenuOptions  Call DisplayText of displayCurrent  Set timeMenuChoice as an int = GetTimeMenuChoice, pass  timeMenuOptions  call PromptAmountTime of Prompts  Instantiate time as a Menu  Set timeTaken as a string = GetValidInt of time  Instantiate timeAssign as an ActivityUtil  Set timeChoice as a string = GetTimeChoice, pass timeMenuChoice,  of timeAssign  Call AssignTime, passed timeChoice, timeTaken, activities, of  timeAssign | Activity  Time Menu |

|  |  |  |
| --- | --- | --- |
| **GetTimeMenuChoice** | | |
| **Input** | **Process** | **Output** |
| timeMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass  timeMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate timeMenu as a Menu  Set timeMenu = a new Menu  Call SetNumOptions of timeMenu, pass length of  timeMenuOptions  Call SetOptions of timeMenu, pass options  Set menuChoice as an int = GetValidMenuChoice of  timeMenu  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **DisplayTicketmenu** | | |
| **Input** | **Process** | **Output** |
| activities | Processing Items:  None  Algorithm:  Clear the console  Instantiate ticketMenuDisplays as a Prompts[], length 1  Instantiate ticketMenuOptions as a MenuDisplays[], length 2  Call SetMenuType of MenuDisplays, pass 8  Instantiate ticketMenu as a DisplaysFile, pass  ticketMenuDisplays and ticketMenuOptions  Call GetMenuDisplay of ticketMenu  Instantiate displayCurrent as a DisplaysReports, pass  ticketMenuDisplays and ticketMenuOptions  Call DisplayText of displayCurrent  Set ticketMenuChoice as an int = GetTicketMenuChoice, pass  ticketMenuOptions  Instantiate ticketAssign as an ActivityUtil  Call AssignTicket, pass ticketMenuChoice and activities, of ticketAssign | Activity  Ticket Menu |

|  |  |  |
| --- | --- | --- |
| **GetTicketMenuChoice** | | |
| **Input** | **Process** | **Output** |
| ticketMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass  ticketMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate ticketMenu as a Menu  Set ticketMenu = a new Menu  Call SetNumOptions of ticketMenu, pass length of  ticketMenuOptions  Call SetOptions of ticketMenu, pass options  Set menuChoice as an int = GetValidMenuChoice of  ticketMenu  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **DisplayEditActMenu** | | |
| **Input** | **Process** | **Output** |
| activities  actEditID | Processing Items:  None  Algorithm:  Clear the console  Instantiate editActMenu as an ActivityFile, pass activities  Call GetAllActivities of editActMenu  Instantiate checkForActivities as an ActivityReport  If CheckIfAnyActivities, passed activities, of checkForActivities < 0  Call DisplayActivityMenu  End if  Call PromptForActIDEdit of Prompts  Input actEditID  Set lineToEdit as an int = GetUserLineToEdit, passed actEditID and  activities, of editActMenu  If lineToEdit < 0  Call AbortActEditMessage of Prompts  Call DisplayActivityMenu  End if  Call EditingMessage of Prompts  Instantiate singleLine as an ActivityReport, pass activities  Call DisplayOneActivity, pass lineToEdit, of singleLine  Call ActEditOrDeleteMenu, pass lineToEdit, activities, and singleLine | Activity  Edit Menu |

|  |  |  |
| --- | --- | --- |
| **ActEditOrDeleteMenu** | | |
| **Input** | **Process** | **Output** |
| lineToEdit  activities  singleLine | Processing Items:  None  Algorithm:  Instantiate actEditMenuDisplays as a Prompts[], length 1  Instantiate actEditMenuOptions as a MenuDisplays[], length 3  Call SetMenuType of MenuDisplays, pass 9  Instantiate editOrDeleteMenu as a DisplaysFile, pass  actEditMenuDisplays and actEditMenuOptions  Call GetMenuDisplay of editOrDeleteMenu  Instantiate displayCurrent as a DisplaysReports, pass  actEditMenuDisplays and actEditMenuOptions  Call DisplayText of displayCurrent  Set editOrDeleteChoice as an int = GetActEditOrDeleteChoice,  Pass actEditMenuOptions  Call RouteActEditOrDelete, pass editOrDeleteChoice, lineToEdit,  activities, and singleLine | Edit/Delete  Menu |

|  |  |  |
| --- | --- | --- |
| **GetActEditOrDeleteChoice** | | |
| **Input** | **Process** | **Output** |
| actEditMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass  actEditMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate actEditOrDelete as a Menu  Set actEditOrDelete = a new Menu  Call SetNumOptions of actEditOrDelete, pass length of  actEditMenuOptions  Call SetOptions of actEditOrDelete, pass options  Set menuChoice as an int = GetValidMenuChoice of  actEditOrDelete  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **RouteActEditOrDelete** | | |
| **Input** | **Process** | **Output** |
| editOrDeleteChoice  lineToEdit  activities  singleLine | Processing Items:  None  Algorithm:  Declare a switch statement, variable editOrDeleteChoice  In case 1: Call DisplayEditActFieldMenu, pass lineToEdit,  activities, and singleLine  In case 2: Call DeleteActivityChoice, pass lineToEdit,  activities, and singleLine  Default to: Call DisplayActivityMenu  End switch | Edit Activity  Field Menu  Delete Activity  Activity Menu |

|  |  |  |
| --- | --- | --- |
| **DisplayEditActFieldMenu** | | |
| **Input** | **Process** | **Output** |
| lineToEdit  activities  singleLine | Processing Items:  None  Algorithm:  Call EditingMessage of Prompts  Instantiate actEditFieldMenuDisplays as a Prompts[], length 1  Instantiate actEditFieldMenuOptions as a MenuDisplays[], length 7  Call SetMenuType of MenuDisplays, pass 10  Instantiate editOrDeleteMenu as a DisplaysFile, pass  actEditFieldMenuDisplays and actEditFieldMenuOptions  Call GetMenuDisplay of editOrDeleteMenu  Instantiate displayCurrent as a DisplaysReports, pass  actEditFieldMenuDisplays and actEditFieldMenuOptions  Call DisplayText of displayCurrent  Set fieldEditChoice as an int = GetEditActFieldChoice,  Pass actEditFieldMenuOptions  Call RouteActEditOrDelete, pass lineToEdit, fieldEditChoice, activities | Edit Activity  Field Menu |

|  |  |  |
| --- | --- | --- |
| **GetEditActFieldChoice** | | |
| **Input** | **Process** | **Output** |
| actEditFieldMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass  actEditFieldMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate editActivityField as a Menu  Set editActivityField = a new Menu  Call SetNumOptions of editActivityField, pass length of  actEditFieldMenuOptions  Call SetOptions of editActivityField, pass options  Set menuChoice as an int = GetValidMenuChoice of  editActivityField  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **EditActivityField** | | |
| **Input** | **Process** | **Output** |
| lineToEdit  fieldEditChoice  activities | Processing Items:  None  Algorithm:  If fieldEditChoice = 7  Call DisplayActivityMenu  Else  Instantiate newEdit as an ActivityUtil, pass activities  Call EditActivityField, pass fieldEditChoice, lineToEdit, and activities  Of newEdit  Instantiate newCount as an ActivityFile, pass activities  Call EditActivityToFile, pass activities, of ActivityFile  Call DisplayActivityMenu  End if | Updated  Activity  Activity  Menu |

|  |  |  |
| --- | --- | --- |
| **DeleteActivityChoice** | | |
| **Input** | **Process** | **Output** |
| lineToEdit  activities  singleLine | Processing Items:  None  Algorithm:  Call DeletingMessage of Prompts  Call DisplayOneActivity, pass lineToEdit, of singleLine  Instantiate deleteActMenuDisplays as a Prompts[], length 1  Instantiate deleteActMenuOptions as a MenuDisplays[], length 2  Call SetMenuType of MenuDisplays, pass 11  Instantiate deleteActMenu as a DisplaysFile, pass  deleteActMenuDisplays and deleteActMenuOptions  Call GetMenuDisplay of deleteActMenu  Instantiate displayCurrent as a DisplaysReports, pass  deleteActMenuDisplays and deleteActMenuOptions  Call DisplayText of displayCurrent  Set deleteVerify as an int = GetDeleteVerify,  Pass deleteActMenuOptions  Call RouteDeleteActChoice, pass deleteVerify, lineToEdit, activities | Delete Activity  Menu |

|  |  |  |
| --- | --- | --- |
| **GetDeleteActVerify** | | |
| **Input** | **Process** | **Output** |
| deleteActMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass  deleteActMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate verifyDeleteActivity as a Menu  Set verifyDeleteActivity = a new Menu  Call SetNumOptions of verifyDeleteActivity, pass length of  deleteActMenuOptions  Call SetOptions of verifyDeleteActivity, pass options  Set menuChoice as an int = GetValidMenuChoice of  verifyDeleteActivity  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **RouteDeleteActChoice** | | |
| **Input** | **Process** | **Output** |
| deleteVerify  lineToEdit  activities  actEditID | Processing Items:  None  Algorithm:  If deleteverify = 1  Call DeleteActivityFromFile, pass lineToEdit and activities,  Of ActivityFile  Instantiate deleteComplete as a new CompleteActivity,  Length [50]  Instantiate deleteActivity as a CompleteActivityFile,  Pass deleteComplete  Call GetCompletedActivities of deleteActivity  Call DeleteCompletionWithActivity, pass actEditID,  deleteComplete, of CompleteActivityFile  Call DeleteActMessage of Prompts  Call DisplayActivityMenu  Else  Call AbortActDeletionMessage of Prompts  Call DisplayActivityMenu  End if | Activity removal  Completion  removal  Activity Menu |

|  |  |  |
| --- | --- | --- |
| **DisplayCompleteMenu** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Clear the console  Instantiate completeMenuDisplays as a Prompts[], length 1  Instantiate completeActMenuOptions as a MenuDisplays[], length 4  Call SetMenuType of MenuDisplays, pass 17  Instantiate deleteActMenu as a DisplaysFile, pass  completeMenuDisplays and completeActMenuOptions  Call GetMenuDisplay of deleteActMenu  Instantiate displayCurrent as a DisplaysReports, pass  completeMenuDisplays and completeActMenuOptions  Call DisplayText of displayCurrent  Set completeChoice as an int = GetCompleteMenuChoice,  Pass completeActMenuOptions  Call RouteCompleteMenu, pass completeChoice | Completion  Menu |

|  |  |  |
| --- | --- | --- |
| **GetCompleteMenuChoice** | | |
| **Input** | **Process** | **Output** |
| completeActMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass  completeActMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate completeMenu as a Menu  Set completeMenu = a new Menu  Call SetNumOptions of completeMenu, pass length of  completeActMenuOptions  Call SetOptions of completeMenu, pass options  Set menuChoice as an int = GetValidMenuChoice of  completeMenu  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **RouteCompleteMenu** | | |
| **Input** | **Process** | **Output** |
| completeChoice | Processing Items:  None  Algorithm:  Declare a switch statement, variable completeChoice  In Case 1: Call CompleteActivities  In Case 2: Call EditCompletions  In Case 3: Call DeleteCompletions  Default to: Clear console  Call DisplayMainMenu  End switch | Complete Activities  Menu  Edit Completions  Menu  Delete Completions  Menu  Main Menu |

|  |  |  |
| --- | --- | --- |
| **CompleteActivities** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Clear the console  Instantiate vacationCount as a Vacation[], length 50  Instantiate checkActivities as an Activity[], length 50  Instantiate completedPrimer1 as an ActivityFile, pass  checkActivities  Call GetAllActivities of completedPrimer1  If GetCount of Activity < 1  Call PromptNoActivities of Prompts  Call MainMenuPrompt of Prompts  Call DisplayMainMenu  End if  Instantiate newCompletes as a CompleteActivity[], length  50  Instantiate fillCompletes as a CompleteActivityFile, pass  newCompletes  Call GetCompletedActivities of fillCompletes  Instantiate checkRemaining as a RemainingActivities[],  Length 50  Instantiate completedPrimer2 as a RemainingActivitiesFile,  pass checkRemaining  Call GetRemainingActivities of completedPrimer2  Instantiate remainingActivities as a  RemainingActivitiesReport  If CheckIfAnyRemaining, pass checkRemaining, of  remainingActivities < 0  Call DisplayMainMenu  End if  Instantiate checkForVacation as a VacationFile, pass  vacationCount  Call GetAllVacations of checkForVacation  If GetCount of Vacation > 0  Instantiate completeActivities as a CompleteActivity[],  Length 50  Instantiate processInfo as a CompleteActivityFile, pass  completeActivities  Call GetCompletedActivities of processInfo  If GetNewProcessInformation of processInfo < 0  Call PromptAbortCompletion of Prompts  Else  Call PromptProcessedCompletion of Prompts  End if  Else  Call PromptCantComplete of Prompts  End if  Call MainMenuPrompt of Prompts  Call DisplayMainMenu | New completed activity |

|  |  |  |
| --- | --- | --- |
| **EditCompletions** | | |
| **Input** | **Process** | **Output** |
| completeEditID | Processing Items:  None  Algorithm:  Clear the console  Instantiate editComplete as a CompleteActivity[], length 50  Instantiate editCompletionMenu as a  CompleteActivityFile, pass editComplete  Call GetCompletedActivities of editCompletionMenu  Instantiate checkForCompletions as a  CompleteActivityReport  If CheckIfAnyActivities, pass editComplete, of  checkForCompletions < 0  Call DisplayMainMenu  End if  Call PromptForCompleteIDEdit of Prompts  Input completeEditID  Set lineToEdit as an int = GetUserLineToEdit, pass  completeEditID and editComplete, of  editCompletionMenu  if lineToEdit < 0  Call AbortCompletionEditMessage of Prompts  Call DisplayCompleteMenu  End if  Call EditingMessage of Prompts  Instantiate singleLine as a CompleteActivityReport, pass  editComplete  Call DisplayOneCompleted, pass lineToEdit, of SingleLine  Call EditCompletionFieldMenu, pass lineToEdit,  editComplete, and singleLine | lineToEdit for completion edits |

|  |  |  |
| --- | --- | --- |
| **EditCompletionFieldMenu** | | |
| **Input** | **Process** | **Output** |
| lineToEdit  editComplete  singleLine | Processing Items:  None  Algorithm:  Instantiate completeEditFieldMenuDisplays as a Prompts[], length 1  Instantiate completeEditFieldMenuOptions as a MenuDisplays[],  length 6  Call SetMenuType of MenuDisplays, pass 18  Instantiate completionFieldMenu as a DisplaysFile, pass  completeEditFieldMenuDisplays and  completeEditFieldMenuOptions  Call GetMenuDisplay of completionFieldMenu  Instantiate displayCurrent as a DisplaysReports, pass  completeEditFieldMenuDisplays and  completeEditFieldMenuOptions  Call DisplayText of displayCurrent  Set fieldEditChoice as an int = GetCompletionField,  Pass completeEditFieldMenuOptions  Call RouteCompletionField, pass lineToEdit, fieldEditChoice,  And editComplete | Edit  Completion  Field  Menu |

|  |  |  |
| --- | --- | --- |
| **GetCompletionField** | | |
| **Input** | **Process** | **Output** |
| completeEditFieldMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[],  pass completeEditFieldMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate editCompletionField as a Menu  Set editCompletionField = a new Menu  Call SetNumOptions of editCompletionField, pass  Length of completeEditFieldMenuOptions  Call SetOptions of editCompletionField, pass  options  Set menuChoice as an int = GetValidMenuChoice of  editCompletionField  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **EditCompletionField** | | |
| **Input** | **Process** | **Output** |
| lineToEdit  fieldEditChoice  editComplete | Processing Items:  None  Algorithm:  If fieldEditChoice = 6  Call DisplayCompleteMenu  Else  Instantiate newEdit as a CompleteActivityUtil, pass editComplete  Call EditCompletionField, pass fieldEditChoice, lineToEdit, and  editComplete, of newEdit  Call EditCompletionToFile, pass editComplete, of  CompleteActivityFile  Call DisplayCompleteMenu  End if | Updated  Completion |

|  |  |  |
| --- | --- | --- |
| **DeleteCompletions** | | |
| **Input** | **Process** | **Output** |
| completeDeleteID | Processing Items:  None  Algorithm:  Instantiate deleteComplete as a CompleteActivity[], length 50  Instantiate deleteCompletionMenu as a CompleteActivityFile,  pass deleteComplete  Call GetCompletedActivities of deleteCompletionMenu  Instantiate checkForCompletions as a CompleteActivityReport  If CheckIfAnyActivities, pass deleteComplete, of  checkForCompletions < 0  Call DisplayMainMenu  End if  Call PromptForCompleteIDDelete of Prompts  Input completeDeleteID  Set lineToEdit as an int = GetUserLineToEdit, pass  completeDeleteID and deleteComplete, of  deleteCompletionMenu  if lineToEdit < 0  Call AbortCompletionDeletionMessage of Prompts  Call DisplayCompleteMenu  End if  Call DeletingMessage of Prompts  Instantiate singleLine as a CompleteActivityReport, pass  deleteComplete  Call DislayOneCompleted, pass lineToEdit, of singleLine  Instantiate competeMenuDeleteDisplays as a Prompts[], length  1  Instantiate completeMenuDeleteOptions as a MenuDisplays[],  Length 2  Call SetMenuType, pass 19, of MenuDisplays  Instantiate deleteMenu as a DisplaysFile, pass  completeMenuDeleteDisplays and  completeMenuDeleteOptions  Call GetmenuDisplay of deleteMenu  Instantiate displayCurrent as a DisplaysReports, pass  completeMenuDeleteDisplays and  completeMenuDeleteOptions  Call DisplayText of displayCurrent  Set deleteVerify as an int = GetDeleteCompleteVerify, pass  completeMenuDeleteOptions  Call RouteDeleteCompleteChoice, pass deleteVerify, lineToEdit,  And deleteComplete | lineToEdit for  completion  Deletions |

|  |  |  |
| --- | --- | --- |
| **GetDeleteCompleteVerify** | | |
| **Input** | **Process** | **Output** |
| completeMenuDeleteOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[],  pass completeMenuDeleteOptions  Set options as a string[] = ToArray of convertOptions  Instantiate deleteCompletion as a Menu  Set deleteCompletion = a new Menu  Call SetNumOptions of deleteCompletion, pass  Length of completeMenuDeleteOptions  Call SetOptions of deleteCompletion, pass  options  Set menuChoice as an int = GetValidMenuChoice of  deleteCompletion  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **RouteDeleteCompleteChoice** | | |
| **Input** | **Process** | **Output** |
| deleteVerify  lineToEdit  deleteComplete | Processing Items:  None  Algorithm:  If deleteVerify = 1  Call DeleteCompletionFromFile, pass lineToEdit and  deleteComplete of CompleteActivityFile  Call DeleteCompletionMessage of Prompts  Else  Call AbortCompletionDeletionMessage of Prompts  End if  Call DisplayCompleteMenu | Completion  Removal  Completion  Menu |

|  |  |  |
| --- | --- | --- |
| **DisplayRemianingMenu** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Clear the console  Instantiate remainActivities as a RemainingActivities[], length 50  Instantiate allQuery as an Activity[], length 50  Instantiate omissionQuery as a CompleteActivity[], length 50  Instantiate actsToCompare as an ActivityFile, pass allQuery  Call GetAllActivities of actsToCompare  Instantiate actsToOmit as a CompleteActivityFile, pass  omissionQuery  Call GetCompletedActivities of actsToOmit  Instantiate actsToKeep as a RemainingActivitesFile, pass  remainActivities  if CheckIfAnyActivities, pass remainActivities, of actsTokeep < 0  Call DisplayMainMenu  End if  If GetRemainingActivities of actsToKeep < 0  Call DisplayMainMenu  End if  Instantiate remainSortMenuDisplays as a Prompts[], length 1  Instantiate remainSortMenuOptions as a MenuDisplays[], length  12  Call SetMenuType, pass 12, of MenuDisplays  Instantiate vacationMenu as a DisplaysFile, pass  remainSortMenuDisplays and remainSortMenuOptions  Call GetMenudisplay of vacationMenu  Instantiate displayCurrent as a DisplaysReports, pass  remainSortMenuDisplays and remainSortMenuOptions  Call DisplayText of displayCurrent  Set remainingSortChoice as an int = GetRemainingSortChoice,  Pass remainSortMenuOptions  Call RouteRemainingSortChoice, pass remainingSortChoice and  remainActivities | Remaining  Menu |

|  |  |  |
| --- | --- | --- |
| **GetRemainingSortChoice** | | |
| **Input** | **Process** | **Output** |
| remainSortMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[],  pass remainSortMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate remainingSortMenu as a Menu  Set remainingSortMenu = a new Menu  Call SetNumOptions of remainingSortMenu, pass  Length of remainSortMenuOptions  Call SetOptions of remainingSortMenu, pass  options  Set menuChoice as an int = GetValidMenuChoice of  remainingSortMenu  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **RouteRemainingSortChoice** | | |
| **Input** | **Process** | **Output** |
| remainingSortChoice  reaminActivities | Processing Items:  None  Algorithm:  Instantiate remaining as a RemainingActivitiesReport  Instantiate sort as a RemainingActivitsUtil, pass  remainActivities  Declare a switch statement, variable  remainingSortChoice  In Case 1: Call DisplayAllRemaining, pass  remainActivities, of remaining  In Case 2: Call SortByCategory, pass  remainingSortChoice, of sort  In Case 3: Call SortByCategory, pass  remainingSortChoice, of sort  In Case 4: Call SortByTime, pass  remainingSortChoice, of sort  In Case 5: Call SortByTime, pass  remainingSortChoice, of sort  In Case 6: Call SortByPrice, pass  remainingSortChoice, of sort  In Case 7: Call SortByPrice, pass  remainingSortChoice, of sort  In Case 8: Call SortByPrice, pass  remainingSortChoice, of sort  In Case 9: Call SortByPrice, pass  remainingSortChoice, of sort  In Case 10: Call SortByTicketNeed, pass  remainingSortChoice, of sort  In Case 11: Call SortByTicketNeed, pass  remainingSortChoice, of sort  Default to: Clear the console  Call DisplayMainMenu  End switch  Call DisplayAllRemaining, pass remainActivities, of  remaining  Call MainMenuPrompt of Prompts  Call DisplayMainMenu | Sorted remaining activities |

|  |  |  |
| --- | --- | --- |
| **DisplayTripReportMenu** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Instantiate tripReportMenuDisplays as a Prompts[], length 1  Instantiate tripReportMenuOptions as a MenuDisplays[],  length 5  Call SetMenuType of MenuDisplays, pass 13  Instantiate reportMenu as a DisplaysFile, pass  tripReportMenuDisplays and  tripReportMenuOptions  Call GetMenuDisplay of reportMenu  Instantiate displayCurrent as a DisplaysReports, pass  tripReportMenuDisplays and  tripReportMenuOptions  Call DisplayText of displayCurrent  Set tripReportChoice as an int = GetTripReportChoice,  Pass tripReportMenuOptions  Call TripReportChoice, pass tripReportChoice | Trip Report  Menu |

|  |  |  |
| --- | --- | --- |
| **GetTripReportChoice** | | |
| **Input** | **Process** | **Output** |
| tripReportMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[],  pass tripReportMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate tripReportMenu as a Menu  Set tripReportMenu = a new Menu  Call SetNumOptions of tripReportMenu, pass  Length of tripReportMenuOptions  Call SetOptions of tripReportMenu, pass  options  Set menuChoice as an int = GetValidMenuChoice of  tripReportMenu  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **RouteTripReportChoice** | | |
| **Input** | **Process** | **Output** |
| tripReportChoice | Processing Items:  None  Algorithm:  Instantiate toFileOption as a SaveToFileMenu  Instantiate events as an EventsCat[], length 50  Instantiate foodDrinks as a FoodDrinkCat[], length 50  Instantiate games as a GamesCat[], length 50  Instantiate museums as a MuseumsCat[], length 50  Instantiate natures as a NatureCat[], length 50  Instantiate nightLives as a NightLifeCat[], length 50  Instantiate shoppings as aShoppingCat[], length 50  Instantiate tours as a ToursCat[], length 50  Instantiate workshops as a WorkshopsCat[], length 50  Declare a switch statement, variable tripReportChoice  In case 1: Call TripSummaryProcess, pass toFileOption  In case 2: Call FavActSummaryProcess, pass toFileOption  In case 3: Call IncompleteSummaryProcess, pass  toFileOption  In case 4: Call RecommendSummaryProcess, pass  toFileOption, events, foodDrinks, games,  museums, natures, nightLives, shoppings,  tours, and workshops  Default to: Call SpendingSummaryProcess, pass  toFileOption, events, foodDrinks, games,  museums, natures, nightLives, shoppings,  tours, and workshops | Trip  Summary  Process  Favorite  Summary  Process  Incomplete  Summary  Process  Recommend  Summary  Process  Spending  Summary  Process |

|  |  |  |
| --- | --- | --- |
| **TripSummaryProcess** | | |
| **Input** | **Process** | **Output** |
| toFileOption | Processing Items:  None  Algorithm:  Instantiate dayByDayCompletions as a TripSummaryByDay[], length 50  Instantiate fillDayByDay as a TripSummaryByDayFile, pass  dayByDayCompletions  if GetDayByDayDecisions of fillDayByDay < 0  Call DisplayMainMenu  End if  Instantiate sort, pass dayByDayCompletions, of TripSummaryByDayUtil  Call SortByDay of sort  Instantiate reportByDay, pass dayByDayCompletions, of  TripSummaryByDayReport  Call DisplayTripSummary of reportDayByDay  If DisplaySaveToFileOption of toFileOption > 0  Call SummaryToFile, pass dayByDayCompletions, of fillDayByDay  End if  Call DisplayMainMenu | Day-by-day  Summary |

|  |  |  |
| --- | --- | --- |
| **FacActSummaryProcess** | | |
| **Input** | **Process** | **Output** |
| toFileOption | Processing Items:  None  Algorithm:  Instantiate favoriteActivities as a FavoriteActivities[], length 50  Instantiate fillFavorites as a FavoriteActivitiesFile, pass  favoriteActivities  if GetFavoriteCompletions of fillFavorites < 0  Call DisplayMainMenu  End if  Instantiate sort, pass favoriteActivities, of FavoriteActivitiesUtil  Call SortByRating of sort  Instantiate reportFavorites, pass favoriteActivities, of  FavoriteActivitiesReport  Call DisplayFavorites of reportFavorites  If DisplaySaveToFileOption of toFileOption > 0  Call FavoritesToFile, pass favoriteActivities, of fillFavorites  End if  Call DisplayMainMenu | Favorites  Summary |

|  |  |  |
| --- | --- | --- |
| **IncompleteSummaryProcess** | | |
| **Input** | **Process** | **Output** |
| toFileOption | Processing Items:  None  Algorithm:  Instantiate incompleteActivities as a IncompleteActivities[], length 50  Instantiate fillIncompletes as a IncompleteActivitiesFile, pass  incompleteActivities  if GetIncompleteActivities of fillIncomplete < 0  Call DisplayMainMenu  End if  Instantiate reportIncompletes, pass incompleteActivities, of  IncompleteActivitiesReport  Call DisplayIncompletes of reportIncompletes  If DisplaySaveToFileOption of toFileOption > 0  Call IncompletesToFile, pass incompleteActivities, of fillIncompletes  End if  Call DisplayMainMenu | Incomplete  Activities  Summary |

|  |  |  |
| --- | --- | --- |
| **RecommendSummaryProcess** | | |
| **Input** | **Process** | **Output** |
| toFileOption  events  foodDrinks  games  museums  natures  nightLives  shoppings  tours  workshops | Processing Items:  None  Algorithm:  Instantiate fillRecommendations as a RecommendationsFile, pass  events, foodDrinks, games, museums, natures, nightLives,  shoppings, tours, workshops  if GetRecommendations of fillRecommendations < 0  Call DisplayMainMenu  End if  Instantiate sort as a RecommendationsUtil, pass events, foodDrinks,  games, museums, natures, nightLives, shoppings, tours,  workshops  Call SortByPrice of sort  Instantiate reportRecommendations, pass events, foodDrinks,  games, museums, natures, nightLives, shoppings, tours,  workshops, of RecommendationsReport  Call DisplayRecommendations of reportRecommendations  If DisplaySaveToFileOption of toFileOption > 0  Call RecommendationsToFile, pass events, foodDrinks, games,  museums, natures, nightLives, shoppings, tours, workshops, of  fillRecommendations  End if  Call DisplayMainMenu | Recommended  Activities  Summary |

|  |  |  |
| --- | --- | --- |
| **SpendingSummaryProcess** | | |
| **Input** | **Process** | **Output** |
| toFileOption  events  foodDrinks  games  museums  natures  nightLives  shoppings  tours  workshops | Processing Items:  None  Algorithm:  Instantiate expenditures as a Spending[], length 50  Instantiate fillExpenditures as a SpendingFile, pass expenditures,  events, foodDrinks, games, museums, natures, nightLives,  shoppings, tours, workshops  if GetExpenditures of fillExpenditures < 0  Call DisplayMainMenu  End if  Instantiate sortAndSum as a SpendingUtil, pass expenditures,  events, foodDrinks, games, museums, natures, nightLives,  shoppings, tours, workshops  Call GetSpendingSum of sortAndSum  Call SortByPrice of sortAndSum  Instantiate reportSpending, pass expenditures, events, foodDrinks,  games, museums, natures, nightLives, shoppings, tours,  workshops, of RecommendationsReport  Call DisplayExpenditures of reportSpending  If DisplaySaveToFileOption of toFileOption > 0  Call SpendingToFile, pass expenditures, of fillExpenditures  End if  Call DisplayMainMenu | Spending  Summary |

|  |  |  |
| --- | --- | --- |
| **GetAllActivities** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Call SetCount, pass 0, of Activity  Open inFile, pass “list.txt”  Set line as a string = ReadLine of inFIle  While line IS NOT null  Set tempArray as a string[] = line split by ‘#’  Set minPrice as a string = 3rd element of tempArray, trim ‘$’ from start  Set maxPrice as a string = 4th element of tempArray, trim ‘$’ from start  Set activities[GetCount of activities] = a new Activity, pass tempArray[0],  tempArray[1], tempArray[2], minPrice, maxPrice, tempArray[5], and  tempArray[6]  Call IncCount of Activity  Set line = ReadLine of inFile  End while  Close inFile | activities |

|  |  |  |
| --- | --- | --- |
| **GenActivityID** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Open inFIle, pass “list.txt”  Set line as a string = ReadLine of inFile  If line IS null  Close inFile  Return “1”  End if  Set checkActivityIDs as a string[] = a new string[], length GetCount of Activity  Set index as an int = 0  While line IS NOT null  Set tempArray as a string[] = line split by ‘#’  Set checkActivityIDs[index] = tempArray[0]  Increment index by 1  Set line = ReadLine of inFile  End while  Close inFile  Set maxID as an int = 0;  Set i as an int = 0  While i < GetCount of Activity  If checkActivityIDs[i] as an int > maxID  Set maxID = checkActivityIDs[i] as an int  End if  Increment i by 1  End while  Return maxID + 1 as a string | New  Activity  ID |

|  |  |  |
| --- | --- | --- |
| **GetUserLineToEdit** | | |
| **Input** | **Process** | **Output** |
| actEditID  activities | Processing Items:  None  Algorithm:  Set verifiedID as a string = VerifyEditID, pass actEditID  Set lineToEdit as an int = CompareID, pass verifiedID, activities  Set lineToEdit = CheckLineValidity, pass verifiedID, lineToEdit, activities  Return lineToEdit | lineToEdit |

|  |  |  |
| --- | --- | --- |
| **VerifyEditID** | | |
| **Input** | **Process** | **Output** |
| actEditID | Processing Items:  None  Algorithm:  Set uerID as an int = 0  Set goodInput as a bool = actEditID attempt parse as int to userID  While goodInput IS NOT true  Clear the console  Call PromptValidIDNoExit of Prompts  Input actEditID  Set goodInput = actEditID attempt parse as int to userID  End while  Return actEditID | actEditID |

|  |  |  |
| --- | --- | --- |
| **CompareID** | | |
| **Input** | **Process** | **Output** |
| verifiedID  activities | Processing Items:  None  Algorithm:  Set newID as an int = 0  Set goodInput as a bool = verifiedID attempt parse as int to newID  If goodInput NOT true  Return -2;  End if  If verifiedID as an int = -1  Return -1  End if  Set first as an int = 0  Set last as an int = GetCount -1 of Activity  Set foundIndex as an int = -2  Set found as a bool = false  Declare middle as an int  Set BINARY\_DIVISOR as a constant int = 2  While found NOT true AND first <= last  Set middle = (first + last)/BINARY\_DIVISOR  If GetActID of activities[middle] = verifiedID  Set found = true  Set foundIndex = middle  Else if GetActID of activities[middle] compared to verifiedID > 0  Set last = middle – 1  Else  Set first = middle + 1  End if  End while  Return foundIndex | foundIndex |

|  |  |  |
| --- | --- | --- |
| **CheckLineValidity** | | |
| **Input** | **Process** | **Output** |
| verifiedID  lineToEdit  activities | Processing Items:  None  Algorithm:  Instantiate reList as an ActivityReport, pass activities  Set goodInput as a bool = true  Set newID as an int = 0  While (lineToEdit < 0 AND lineToEdit NOT = -1) OR goodInput NOT true  Clear the console  Call DisplayAllActivities, pass activities, of reList  Call PromptValidIDNoExit of Prompts  Input activityEditID  Set goodInput = activityEditID attempt parse as int to newID  Set lineToEdit = CompareID, pass activityEditID, activities  End while  Return lineToEdit | lineToEdit |

|  |  |  |
| --- | --- | --- |
| **NewActivityToFile** | | |
| **Input** | **Process** | **Output** |
| activities | Processing Items:  None  Algorithm:  Open outFile, pass “list.txt”, in append mode  Write GetConcatActivity of activities[GetCount of Activity] to outFile  Close outFile | Activity  Output  To file |

|  |  |  |
| --- | --- | --- |
| **EditActivityToFile** | | |
| **Input** | **Process** | **Output** |
| activities | Processing Items:  None  Algorithm:  Open outFile, pass “list.txt”  Set i as an int = 0  While i < GetCount of Activity  Write GetConcatActivity of activities[i] to outFile  Increment i by 1  End while  Close outFile | Activity  Output  To file |

|  |  |  |
| --- | --- | --- |
| **DeleteActivityFromFile** | | |
| **Input** | **Process** | **Output** |
| lineToEdit  activities | Processing Items:  None  Algorithm:  Open outFile, pass “list.txt”  Set i as an int = 0  While i < GetCount of Activity  If i = lineToEdit  Continue to next iteration  Else  Write GetConcatActivity of activities[GetCount of Activity] to outFile  End if  Increment i by 1  End while  Close outFile | Activity  Output  To file |

|  |  |  |
| --- | --- | --- |
| **DeleteAllActivitiesFromFile** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Open outFile, pass “list.txt”  Write an empty string to outFile  Close outFile | Activity  Output  To file |

|  |  |  |
| --- | --- | --- |
| **AssignCategory** | | |
| **Input** | **Process** | **Output** |
| catChoice  activities | Processing Items:  None  Algorithm:  Call SetCategory, pass AssignCatLanguage, pass catChoice, of Prompts, of  Activities[GetCount of Activity] | Category  Assignment |

|  |  |  |
| --- | --- | --- |
| **GetValidInteger** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Set number as an int = 0  Input userNumber  Set goodInput as a bool = userNumber attempt parse as int to number  While goodInput IS NOT true  Call PromptValidInt of Prompts  Input userNumber  Set goodInput = userNumber attempt parse as int to number  End while  Return number as a string | Valid integer |

|  |  |  |
| --- | --- | --- |
| **CheckMaxPrice** | | |
| **Input** | **Process** | **Output** |
| userMax  activities | Processing Items:  None  Algorithm:  Set goodInput as a bool = true  Set tempNum as an int = 0  While userMax as an int <  (GetPriceMin of activities[GetCount of Activity] as an int)  Call PromptValidMaxPrice of Prompts  Input userMax  Set goodInput = userMax attempt parse as int to tempNum  While goodInput IS NOT true  Call PromptValidInt of Prompts  Input userMax  Set goodInput = userMax attempt parse as int to tempNum  End while  End while  Return userMax | userMax |

|  |  |  |
| --- | --- | --- |
| **AssignTime** | | |
| **Input** | **Process** | **Output** |
| timeChoice  timeTaken  activities | Processing Items:  None  Algorithm:  If timeTaken as an int > 1  append timeChoice with “s”  end if  Call SetTimeNeeded, pass “timeTake-timeChoice”, of  activities[GetCount of Activity] | Time  Assignment |

|  |  |  |
| --- | --- | --- |
| **GetTimeChoice** | | |
| **Input** | **Process** | **Output** |
| timeMenuChoice | Processing Items:  None  Algorithm:  Set timeChoice as a string[] = a new string[], length 1  Declare a switch statement, variable timeMenuChoice  In Case 1: set timeChoice[0] = AssignMinuteLanguage of  Prompts  In Case 2: set timeChoice[0] = AssignHourLanguage of Prompts  Default to: set timeChoice[0] = AssignDayLanguage of Prompts  End switch  Return timeChoice[0] | Time Choice |

|  |  |  |
| --- | --- | --- |
| **AssignTicket** | | |
| **Input** | **Process** | **Output** |
| ticketMenuChoice  activities | Processing Items:  None  Algorithm:  If ticketMenuChoice = 1  Call SetTicketNeeded, pass AssignYesLanguage of Prompts, of  Activities[GetCount of Activity]  Else  Call SetTicketNeeded, pass AssignNoLanguage of Prompts, of  Activities[GetCount of Activity]  End if | Ticket  Assignment |

|  |  |  |
| --- | --- | --- |
| **EditActivityField** | | |
| **Input** | **Process** | **Output** |
| fieldEditChoice  lineToEdit  activities  newField | Processing Items:  None  Algorithm:  Declare a switch statement, variable fieldEditChoice  In Case 1: Call PrompEnterName of Prompts  Input newField  If newField as lower case = “stop”  Break switch  Else  Call SetName, pass newField, of  activities[lineToEdit]  Break switch  End if  In Case 2: Instantiate catMenuEditDisplays as a Prompts[],  Length 1  Instantiate catMenuEditOptions as a MenuDisplays[],  Length 9  Call SetMenuType, pass 6, of MenuDisplays  Instantiate activityMenu as a DisplaysFile, pass  catMenuEditDisplays, catMenuEditOptions  Call GetMenuDisplay of activityMenu  Instantiate displayCurrent as a DisplaysReports,  Pass catMenuEditDisplays, catMenuEditOptions  Call DisplayText of displayCurrent  Instantiate convertOptions as a MenuDisplayUtil,  Pass catMenuEditOptions  Set options as a string[] = ToArray of convertOptions  Instantiate categoryMenu as a Menu  Set categoryMenu = a new Menu  Call SetNumOptions, pass  Length of catMenuEditOptions of categoryMenu  Call SetOptions, pass options, of categoryMenu  Call AssignCategory, pass GetValidMenuChoice of  categoryMenu, activities  Break switch  In Case 3: Call PromptMinPriceEdit of Prompts  Input newField  If newField as lower case = “stop”  Break switch  Else  Set goodInput as a bool = false  Set tempNum as an int = 0  Set goodInput = newField attempt to parse as int  To tempNum  While newField as an int > GetPriceMax of  activities[lintToEdit] as an int  Call PromptValidMinPrice of Prompts  Input newField  While goodInput IS NOT true  Call PromptValidMinPrice of Prompts  Input newField  Set goodInput = newField attempt parse as int  To tempNum  End while  End while  Call SetPriceMin, pass newField, of  activities[lineToEdit]  End if  Break switch  In Case 4: Call PromptMaxPriceEdit of Prompts  Input newField  If newField as lower case = “stop”  Break switch  Else  Set goodInput as a bool = false  Set tempNum as an int = 0  Set goodInput = newField attempt to parse as int  To tempNum  While newField as an int > GetPriceMin of  activities[lintToEdit] as an int  Call PromptValidMaxPrice of Prompts  Input newField  While goodInput IS NOT true  Call PromptValidMaxPrice of Prompts  Input newField  Set goodInput = newField attempt parse as int  To tempNum  End while  End while  Call SetPriceMax, pass newField, of  activities[lineToEdit]  End if  Break switch  In Case 5: Instantiate editTimeMenuDisplays as a Prompts[],  Length 1  Instantiate editTimeMenuOptions as a  MenuDisplays[], Length 3  Call SetMenuType, pass 7, of MenuDisplays  Instantiate pullTimeMenu as a DisplaysFile, pass  editTimeMenuDisplays, editTimeMenuOptions  Call GetMenuDisplay of pullTimeMenu  Instantiate displayTimePrompt as a DisplaysReports,  Pass editTimeMenuDisplays, editTimeMenuOptions  Call DisplayText of displayTimePrompt  Instantiate convertTimeOptions as a  MenuDisplayUtil, pass editTimeMenuOptions  Set timeOptions as a string[] = ToArray of  convertTimeOptions  Instantiate timeMenu as a Menu  Set timeMenu = a new Menu  Call SetNumOptions, pass  Length of editTimeMenuOptions of timeMenu  Call SetOptions, pass timeOptions, of timeMenu  Call AssignCategory, pass GetValidMenuChoice of  timeMenu, activities  Call PromptAmountTime of Prompts  Instantiate time as a Menu  Set timeTake as a string = GetValidInt of time  Set timeChoice as a string = GetTimeChoice, pass  timeMenuChoice  Call AssignTime, pass timeChoice, timeTaken,  activities  Break switch  In Case 6: Instantiate editTicketMenuDisplays as a Prompts[],  Length 1  Instantiate editTicketMenuOptions as a  MenuDisplays[], Length 2  Call SetMenuType, pass 8, of MenuDisplays  Instantiate pullTicketMenu as a DisplaysFile, pass  editTicketMenuDisplays, editTicketMenuOptions  Call GetMenuDisplay of pullTicketMenu  Instantiate displayTicketPrompt as a DisplaysReports,  Pass editTicketMenuDisplays,  editTicketMenuOptions  Call DisplayText of displayTicketPrompt  Instantiate convertTicketOptions as a  MenuDisplayUtil, pass editTicketMenuOptions  Set ticketOptions as a string[] = ToArray of  convertTicketOptions  Instantiate ticketMenu as a Menu  Set ticketMenu = a new Menu  Call SetNumOptions, pass  Length of editTicketMenuOptions of ticketMenu  Call SetOptions, pass ticketOptions, of ticketMenu  Call AssignCategory, pass GetValidMenuChoice of  ticketMenu, activities  Break switch | Updated  Activity  Details |

|  |  |  |
| --- | --- | --- |
| **GetCompletedActivities** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Call SetCount, pass 0, of CompleteActivity  Open inFile, pass “completed.txt”  Set line as a string = ReadLine of inFIle  While line IS NOT null  Set tempArray as a string[] = line split by ‘#’  Set amountSpent as a string = 4th element of tempArray,  trim ‘$’ from start  Set CompleteActivities[GetCount of CompleteActivity] =  a new Activity, pass tempArray[0], tempArray[1],  tempArray[2], tempArray[3], amountSpent, tempArray[5],  tempArray[6], tempArray[7], tempArray[8]  Call IncCount of CompleteActivity  Set line = ReadLine of inFile  End while  Close inFile | Completed  Activities |

|  |  |  |
| --- | --- | --- |
| **GetCompletedActivities** | | |
| **Input** | **Process** | **Output** |
| processInput | Processing Items:  None  Algorithm:  Instantiate processedActivities as a CompleteActivity[], length 50  Set processedActivities[GetCount of CompleteActivity] =  A new CompleteActivity  Call SetCompleteID, pass GenCompleteID, of  processedActivities[GetCount of CompleteActivity]  Call PromptForActIDComplete of Prompts  Instantiate completeMenu as a CompleteActivityMenu  Set completeID as a string = GetValidID of completeMenu  If completeID = “-1”  Return -1  End if  Call SetName, pass GetCompleteName, pass completeID, of  processedActivities[GetCount of CompleteActivity]  Call SetOriginID, pass completeID, of  processedActivities[GetCount of CompleteActivity]  Call PromptDateCompleted of Prompts  Input processInput  Instantiate newProcess as a CompleteActivityUtil  Set processInput = ResolveDateErrors, pass processInput, of  newProcess  Call SetDateComplete, pass processInput, of  processActivityies[GetCount of CompleteActivity]  Call PromptExpenditures of Prompts  Input processInput  Call SetMoneySpent, pass “$” + GetValidSpent,  Pass processInput, of newProcess, of  processedActivities[GetCount of CompleteActivity]  Instantiate processedRating as a CompleteActivityMenu  Call SetRating, pass ProcessRatingMenu of processedRating, of  processedActivities[GetCount of CompleteActivity]  Call PromptReview of Prompts  Input processInput  Call SetReview, pass GetValidReview, pass processInput, of  newProcess, of  processedActivities[GetCount of CompleteActivity]  Instantiate processedRecommend as a CompleteActivityMenu  Call SetRecommended, pass ProcessRecommendMenu, of  processedRating, of  processedActivities[GetCount of CompleteActivity]  Call SetCategory, pass GetCompleteCategory, pass completeID,  Of processedActivities[GetCount of CompleteActivity]  Call NewActivityToCompletion, pass processedActivities  Call IncCount of CompleteActivity  Return 1 | New  Completed  Activities |

|  |  |  |
| --- | --- | --- |
| **GetUserLineToEdit** | | |
| **Input** | **Process** | **Output** |
| completeEditID editComplete | Processing Items:  None  Algorithm:  Set verifiedID as a string = VerifyEditID, pass completeEditID  Set lineToEdit as an int = CompareID, pass verifiedID,  editComplete  Set lineToEdit = CheckLineValidity, pass verifiedID, lineToEdit,  CompleteActivities  Return completeEditID | completeEditID |

|  |  |  |
| --- | --- | --- |
| **VerifyEditID** | | |
| **Input** | **Process** | **Output** |
| completeEditID | Processing Items:  None  Algorithm:  Set uerID as an int = 0  Set goodInput as a bool = completeEditID attempt parse as int to  userID  While goodInput IS NOT true  Clear the console  Call PromptValidIDNoExit of Prompts  Input completeEditID  Set goodInput = completeEditID attempt parse as int to userID  End while  Return completeEditID | completeEditID |

|  |  |  |
| --- | --- | --- |
| **CompareID** | | |
| **Input** | **Process** | **Output** |
| verifiedID  editComplete | Processing Items:  None  Algorithm:  If verifiedID as an int = -1  Return -1  End if  Set i as an int = 0  While i < GetCount of CompleteActivity  If verifiedID = GetCompleteID of editComplete[i]  Return i  End if  Increment i by 1  End while  Return -2 | ID line |

|  |  |  |
| --- | --- | --- |
| **CheckLineValidity** | | |
| **Input** | **Process** | **Output** |
| verifiedID  lineToEdit  editComplete | Processing Items:  None  Algorithm:  Instantiate reList as an ActivityReport, pass editComplete  Set goodInput as a bool = true  Set newID as an int = 0  While (lineToEdit < 0 AND lineToEdit NOT = -1) OR goodInput NOT true  Clear the console  Call DisplayAllActivities, pass editComplete, of reList  Call PromptValidIDNoExit of Prompts  Input completeEditID  Set goodInput = completeEditID attempt parse as int to newID  Set lineToEdit = CompareID, pass completeEditID, editComplete  End while  Return lineToEdit | lineToEdit |

|  |  |  |
| --- | --- | --- |
| **GetCompleteName** | | |
| **Input** | **Process** | **Output** |
| completeID | Processing Items:  None  Algorithm:  Open inFile, pass “list.txt”  Set line as a string = ReadLine of inFile  While line IS NOT null  Set tempArray as a string[] = line split by ‘#’  If completeID = tempArray[0]  Close inFile  Return tempArray[1]  End if  Set line = ReadLine of inFile  End while  Close inFile  Return “NAME\_NOT\_FOUND” | Activity  name |

|  |  |  |
| --- | --- | --- |
| **GenCompleteID** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Open inFIle, pass “completed.txt”  Set line as a string = ReadLine of inFile  If line IS null  Close inFile  Return “1”  End if  Set checkCompleteIDs as a string[] = a new string[], length GetCount of  CompleteActivity  Set index as an int = 0  While line IS NOT null  Set tempArray as a string[] = line split by ‘#’  Set checkCompleteIDs [index] = tempArray[0]  Increment index by 1  Set line = ReadLine of inFile  End while  Close inFile  Set maxID as an int = 0;  Set i as an int = 0  While i < GetCount of CompleteActivity  If checkCompleteIDs[i] as an int > maxID  Set maxID = checkCompleteIDs[i] as an int  End if  Increment i by 1  End while  Return maxID + 1 as a string | New  Complete  ID |

|  |  |  |
| --- | --- | --- |
| **GetCompleteCategory** | | |
| **Input** | **Process** | **Output** |
| completeID | Processing Items:  None  Algorithm:  Open inFile, pass “list.txt”  Set line as a string = ReadLine of inFile  While line IS NOT null  Set tempArray as a string[] = line split by ‘#’  If tempArray[0] = completeID  Close inFile  Return tempArray[2]  End if  Set line = ReadLine of inFile  End while  Close inFile | Category  For new  completions |

|  |  |  |
| --- | --- | --- |
| **NewActivityToCompletion** | | |
| **Input** | **Process** | **Output** |
| Processed  Activities | Processing Items:  None  Algorithm:  Open outFile, pass “completed.txt”, in append mode  Write GetConcatCompleted of  processedActivities[GetCount of CompleteActivity] to outFile  Close outFile | Completion  Output  To file |

|  |  |  |
| --- | --- | --- |
| **EditCompletionToFile** | | |
| **Input** | **Process** | **Output** |
| Processed  Activities | Processing Items:  None  Algorithm:  Open outFile, pass “completed.txt”  Set i as an int = 0  While i < GetCount of CompleteActivity  Write GetConcatCompleted of editComplete[i] to outFile  Increment i by 1  End while  Close outFile | Completion  Output  To file |

|  |  |  |
| --- | --- | --- |
| **DeleteCompletionFromFile** | | |
| **Input** | **Process** | **Output** |
| lineToEdit  delete  Complete | Processing Items:  None  Algorithm:  Open outFile, pass “completed.txt”  Set i as an int = 0  While i < GetCount of CompletedActivity  If i = lineToEdit  Continue to next iteration  Else  Write GetConcatActivity of deleteComplete[GetCount of Activity] to  outFile  End if  Increment i by 1  End while  Close outFile | Completion  Removal  From file |

|  |  |  |
| --- | --- | --- |
| **DeleteCompletionWithActivity** | | |
| **Input** | **Process** | **Output** |
| originID  delete  Complete | Processing Items:  None  Algorithm:  Open outFile, pass “completed.txt”  Set i as an int = 0  While i < GetCount of CompletedActivity  If GetOriginID of deleteComplete[i] = originID  Continue to next iteration  Else  Write GetConcatActivity of deleteComplete[GetCount of Activity] to  outFile  End if  Increment i by 1  End while  Close outFile | Completion  Removal  From file |

|  |  |  |
| --- | --- | --- |
| **DeleteAllCompletionsFromFile** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Open outFile, pass “completed.txt”  Write an empty string to outFile  Close outFile | Completion  Output  To file |

|  |  |  |
| --- | --- | --- |
| **GetValidID** | | |
| **Input** | **Process** | **Output** |
| number | Processing Items:  None  Algorithm:  Set userNum as an int = 0  Input number  If number as lower case = “stop”  Return “-1”  End if  Set goodInput as a bool = number attempt parse as int to userNum  While goodInput IS NOT true  Call promptValidID of Prompts  Input number  If number as lower case = “stop”  Return “-1”  End if  Set goodInput = number attempt parse as int to userNum  End while  Return userNum as a string | Valid ID |

|  |  |  |
| --- | --- | --- |
| **ProcessRatingMenu** | | |
| **Input** | **Process** | **Output** |
| Selection | Processing Items:  None  Algorithm:  Instantiate ratingMenuDisplays as a Prompts[], length 1  Instantiate ratingMenuOptions as a MenuDisplays[], length 5  Call SetMenuType, pass 15, of MenuDisplays  Instantiate ratingMenu as a DisplaysFile, pass  ratingMenuDisplays, ratingMenuOptions  Call GetMenuDisplay of ratingMenu  Instantiate displayCurrent as a DisplaysReports, pass  ratingMenuDisplays, ratingMenuOptions  Call DisplayText of displayCurrent  Return AssignRating, pass GetProcessRating, pass ratingMenuOptions, of  Prompts | Rating |

|  |  |  |
| --- | --- | --- |
| **GetProcessRating** | | |
| **Input** | **Process** | **Output** |
| ratingMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass  ratingMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate processRatingMenu as a Menu  Set processRatingMenu = a new Menu  Call SetNumOptions of processRatingMenu, pass length of  ratingMenuOptions  Call SetOptions of processRatingMenu, pass options  Set menuChoice as an int = GetValidMenuChoice of  processRatingMenu  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **ProcessRecommendMenu** | | |
| **Input** | **Process** | **Output** |
| Selection | Processing Items:  None  Algorithm:  Instantiate recommendMenuDisplays as a Prompts[], length 1  Instantiate recommendMenuOptions as a MenuDisplays[], length 2  Call SetMenuType, pass 16, of MenuDisplays  Instantiate recommendMenu as a DisplaysFile, pass  recommendMenuDisplays, recommendMenuOptions  Call GetMenuDisplay of recommendMenu  Instantiate displayCurrent as a DisplaysReports, pass  recommendMenuDisplays, recommendMenuOptions  Call DisplayText of displayCurrent  Return AssignRecommend, pass GetProcessRecommendation, pass recommendMenuOptions, of Prompts | Recommendation |

|  |  |  |
| --- | --- | --- |
| **GetProcessRecommendation** | | |
| **Input** | **Process** | **Output** |
| recommendMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil[], pass  recommendMenuOptions  Set options as a string[] = ToArray of convertOptions  Instantiate processRecommendMenu as a Menu  Set processRecommendMenu = a new Menu  Call SetNumOptions of processRecommendMenu, pass  length of recommendMenuOptions  Call SetOptions of processRecommendMenu, pass  options  Set menuChoice as an int = GetValidMenuChoice of  processRecommendMenu  Return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **ResolveDateErrors** | | |
| **Input** | **Process** | **Output** |
| badDate | Processing Items:  None  Algorithm:  Set dateArray as a string[] = badDate split by “/”  Set monthSpecifier as an int = 0  Set daysInMonths as an int[] = 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31  Open vacationFile, pass “vacation.txt”  Set vacationLine as a string = ReadLine of vacationFile  Set vacationArray as a string[] = vacationLine split by ‘#’  Set vacationStart as a string = vacationArray[2]  Set vacationStartArray as a string[] = vacationStart split by “/”  Set vacationStartMonth as an int = vacationStartArray[0] as an int  Set vacationStartDay as an int = vacationStartArray[1] as an int  Set vacationStartYear as an int = vacationStartArray[02 as an int  Set vacationEnd as a string = vacationArray[3]  Set vacationEndArray as a string[] = vacationEnd split by “/”  Set vacationEndMonth as an int = vacationEndArray[0] as an int  Set vacationEndDay as an int = vacationEndArray[1] as an int  Set vacationEndYear as an int = vacationEndArray[02 as an int  Close vacationFile  Set i as an int = 0  While i < 3  Set failedCheck as a bool = false  Set number as an int = 0  Set notADay as a bool = false  Set userNumber as a string = dateArray[i]  Set goodInput as a bool = userNumber attempt parse as int to number  If i = 0  Set monthSpecifier = number  End if  If i = 1  If number < 1 OR number > daysInMonths[monthSpecifier -1 ]  Set notADay = true  End if  End if  While goodInput IS NOT true OR length of dataArray < 3 OR length of  dateArray > 3 OR (i = 0 AND (number < 1 OR number > 12)) OR  notADay IS true OR (i = 2 AND (number < 1999 OR number > 2099))  Call PrompDateError, pass 4, -1, of Prompts  Input badDate  Set dateArray = badDate split by “/”  Set j as an int = 0  While j < 3  If length of dateArray < 3 OR length of dateArray > 3  Break loop  End if  Set goodInput = dateArray[j] attempt parse as int to number  If goodInput IS NOT true  Break loop  End if  Increment j by 1  End while  Set notADay = false  Set i = -1  End while  Set completeYear as an int = dateArray[2] as an int  If i = 0 AND number < vacationStartMonth AND completeYear <=  vacationStartYear  set failedCheck = true  set i = -1  else if i = 0 AND number > vacationEndMonth AND completeYear >=  vacationEndYear  set failedCheck = true  set i = -1  end if  if i = 1 AND number < vacationStartDay AND monthSpecifier <=  vacationStartMonth AND completeYear <= vacationStartYear  set failedCheck = true  set i = -1  else if i = 1 AND number > vacationEndDay AND monthSpecifier >=  vacationEndMonth AND completeYear >= vacationEndYear  set failedCheck = true  set i = -1  end if  if i = 2  if number < vacationStartYear OR number > vacationEndYear  set failedCheck = true  set i = -1  end if  if failedCheck IS true  Call PromptCompleteDateError, pass vacationStart, vacationEnd  Of Prompts  Input badDate  Set dateArray = badDate split by “/”  End if  Increment i by 1  Return badDate | badDate |

|  |  |  |
| --- | --- | --- |
| **GetValidSpent** | | |
| **Input** | **Process** | **Output** |
| spent | Processing Items:  None  Algorithm:  Set goodSpent as an int = 0  Set goodInput as a bool = spent attempt parse as int to goodSpent  While goodInput IS NOT true OR goodSpent < 0  Call PromptValidSpent of Prompts  Input spent  Set goodInput = spent attempt parse as into to goodSpent  End while  Return spent | spent |

|  |  |  |
| --- | --- | --- |
| **GetValidReview** | | |
| **Input** | **Process** | **Output** |
| review | Processing Items:  None  Algorithm:  If length of review > 0  Return review  Else  Return GetPlaceHolderReview of Prompts  End if | Review  Prompt |

|  |  |  |
| --- | --- | --- |
| **EditCompletionField** | | |
| **Input** | **Process** | **Output** |
| fieldEditChoice  lineToEdit  editComplete | Processing Items:  None  Algorithm:  Clear the console  Declare a switch statement, variable fieldEditChoice  In Case 1: Call PromptDateCompleted of Prompts  Input newField  If newFIeld as lower case = “stop”  Break switch  Else  Call SetDateComplete, pass ResolveDateErrors, pass  newField, of editComplete[lineToEdit]  break switch  In Case 2: Call PromptExpenditures of Prompts  Input newField  If newFIeld as lower case = “stop”  Break switch  Else  Call SetMoneySpent, pass GetValidSpent, pass  newField, of editComplete[lineToEdit]  break switch  In Case 3: Instantiate ratingMenuEditDisplays as a Prompts,  Length 1  Instantiate ratingMenuEditOptions as a MenuDisplays,  Length 5  Call SetMenuType, pass 15, of MenuDisplays  Instantiate activityMenu as a DisplaysFile, pass  ratingMenuEditDisplays, ratingMenuEditOptions  Call GetMenuDisplay of activityMenu  Instantiate displayCurrent as a DisplaysReports, pass  ratingMenuEditDisplays, ratingMenuEditOptions  Call DisplayText of displayCurrent  Instantiate convertOptions as a MenuDisplaysUtil, pass  ratingMenuEditOptions  set options as a string[] = ToArray of convertOptions  Instantiate ratingMenu as a Menu  Set ratingMenu = a new Menu  Call SetNumOptions, pass length of  ratingMenuEditOptions, of ratingMenu  Call SetOptions, pass options, of ratingMenu  Call SetRating, pass AssignRating, pass  GetValidMenuChoice, of ratingMenu, of Prompts, of  editComplete[lineToEdit]  break switch  In Case 4: Call PromptReview of Prompts  Input newField  If newFIeld as lower case = “stop”  Break switch  Else  Call SetReview, pass GetValidReview, pass  newField, of editComplete[lineToEdit]  break switch  Default to: Instantiate recommendMenuEditDisplays as a Prompts,  Length 1  Instantiate recommendMenuEditOptions as a  MenuDisplays, Length 5  Call SetMenuType, pass 16, of MenuDisplays  Instantiate recommendMenu as a DisplaysFile, pass  recommendMenuEditDisplays,  recommendMenuEditOptions  Call GetMenuDisplay of recommendMenu  Instantiate displayRecommend as a DisplaysReports,  Pass recommendMenuEditDisplays,  recommendMenuEditOptions  Call DisplayText of displayRecommend  Instantiate convertRecommendOptions as a  MenuDisplaysUtil, pass recommendMenuEditOptions  set options as a string[] = ToArray of  convertRecommendOptions  Instantiate recommendMenu as a Menu  Set recommendMenu = a new Menu  Call SetNumOptions, pass length of  recommendMenuEditOptions, of recommendMenu  Call SetOptions, pass options, of recommendMenu  Call SetRating, pass AssignRating, pass  GetValidMenuChoice, of recommendMenu, of  Prompts, of editComplete[lineToEdit]  break switch | Completion  Field  edits |

|  |  |  |
| --- | --- | --- |
| **GetMenuDisplay** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Call SetCount, pass 0, of Prompts  Call SetCOunt, pass 0, of MenuDisplays  Set fileToOpen as a string = GetMenuFilePath  Open inFile, pass fileToOpen  Set line as a string = ReadLine of inFile  While line IS NOT null  Set tempArray as a string[] = line split by ‘#’  If tempArray[0] as an int = GetLanguageID of Prompts  If tempArray[1] as an int = GetDisplayID of Prompts  Set displays[GetCount of Prompts] = a Prompts, pass tempArray[2]  Call IncCount of Prompts  Else if tempArray[1] as an int = GetoptionsID of Prompts  Set menuOptions[GetCount of MenuDisplays] = a MenuDisplays, pass  tempArray[2]  Call IncCount of MenuDisplays  End if  End if  Set line = ReadLine of inFile  End while  Close inFile | Menus |

|  |  |  |
| --- | --- | --- |
| **GetMenuDisplay** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Open pathIn, pass “MenuFilePaths.txt”  Set line as a string = ReadLine of pathIn  Set count as an int = 0  While line IS NOT null  If count NOT EQUAL to GetMenuType of MenuDisplays  Increment count by 1  Set line = ReadLine of pathIn  Else  Break loop  End if  End while  Close inFile  Return line | Menu file path |

|  |  |  |
| --- | --- | --- |
| **GetFavoriteCompletions** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Call SetCount, pass 0, of FavoriteActivities  Open completedFile, pass”completed.txt”  Set completedLine as a string = ReadLine of completedFile  If completeLine IS null  Call NoneCompleted of Prompts  Close completedFile  Return -1  End if  While completeLine IS NOT null  Set tempArray as a string[] = completeLine split by ‘#’  Set ratingArray as a string[] = tempArray[5] split by “ “  If ratingArray[0] as an int < 4  Set completeLine = ReadLine of completedFile  Continue to next iteration  Else  Set favoriteActivities[GetCount of FavoriteActivities] = a favoriteActivities, pass  tempArray[0], tempArray[2], tempArray[5]  Call IncCount of FavoriteActivities  Set completeLine = ReadLine of completedFile  End if  End while  Close completedFile  Return 1 | Favorites |

|  |  |  |
| --- | --- | --- |
| **SortByRating** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Set i as an int = 0  While i < GetCount – 1 of FavoriteActivities  Set max as an int = i  Set j as an int = i + 1  While j < GetCount of FavoriteActivities  Set maxRatingArray as a string[] = GetRating of favoriteActivities[max] split  By “ “  Set nextRatingArray as a string[] = GetRating of favoriteActivities[j] split  By “ “  If nextRatingArray[0] as an int > maxRatingArray[0] as an int  Set max = j  End if  Increment j by 1  End while  Increment i by 1  End while | Sorted  Favorites |

|  |  |  |
| --- | --- | --- |
| **SortByRating** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as a FavoriteActivities = favoriteActivities[x]  Set favoriteActivities[x] = favoriteActivities[y]  Set favoriteActivities[y] = temp | Swapped  favorites |

|  |  |  |
| --- | --- | --- |
| **GetIncompleteActivities** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Set allActivities as a string[,] = GetAllActivities  Set omissionIDs as a string[] = GetOmissionIDs  Set clearOmitIDs as a string[] = a new string, size GetCount of Activity  Call SetCount, pass 0, of IncompleteActiviies  If allActivities[0,0] = “-1”  Call NoneIncompleted of Prompts  Return -1  End if  If omissionIDs[0] = “-1”  Set i as an int = 0  While i < GetCount of Activity  Set incompleteActivities[GetCount of IncompleteActiviites] =  A new IncompleteActivities, pass allActivities[i,0], allActivities[i,1]  Call IncCount of IncompleteActivities  Increment i by 1  End while  Return 1  End if  If CompareAllActivities, pass allActivities, omissionIDs, < 0  Call NoneIncompleted of Prompts  Return -1  End if  Set i as an int = 0  While i < GetCount of Activity  Set j as an int = 0  While j < GetCount of CompleteActivity  If omissionIDs[j] = allActvities[i,0]  Break loop  Else if omissionIDs[j] NOT EQUAL to allActivities[i,0] AND  j NOT EQUAL to GetCount -1 of CompleteActivity  continue to next iteration  else  set incompleteActivities[GetCount of IncompleteActivities] = a new  IncompleteActivities, pass allActivities[i,0], allActivities[i,1]  Call IncCount of IncompleteActivities  Set clearOmitIDs[i] = allActivities[i,0]  End if  Increment j by 1  End while  Increment i by 1  End while  Return 1 | Incomplete  Activities |

|  |  |  |
| --- | --- | --- |
| **GetAllActivities** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Set count as an int = 0  Instantiate forComparision as an Activity[], length 50  Instantiate countActvities as an ActivitiyFile, pass forComparison  Call GetAllActivities of countActivities  Set allActivities as a string[,] = a new string, length [GetCount of Activity, 7]  Open inFile, pass “list.txt”  Set line as a string = ReadLine of inFile  If line IS null  Set noActivities as a string[,] = a new string, length [1, 7]  Set noActivities[0,0] = “-1”  Close inFile  Return noActivities  End if  While line IS NOT null  Set tempArray as a string[] = line split by ‘#’  Set i as an int = 0  While i < 7  Set allActivities[count, i] = tempArray[i]  Increment i by 1  End while  Increment count by 1  Set line = ReadLine of inFile  End while  Close inFile  Return allActivities | All activities |

|  |  |  |
| --- | --- | --- |
| **GetOmissionIDs** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Set count as an int = 0  Instantiate forComparision as a CompleteActivity[], length 50  Instantiate countCompletes as a CompleteActivityFile, pass forComparison  Call GetCompletedActivities of countCompletes  Set omissionIDs = a new string[], length GetCount of CompleteActivity  Open inFile, pass “completed.txt”  Set line as a string = ReadLine of inFile  If line IS null  Set noneCompleted as a string[] = {“-1”}  Close inFile  Return noneCompleted  End if  While line IS NOT null  Set tempArray as a string[] = line split by ‘#’  Set omissionIDs[count] = tempArray[1]  Increment count by 1  Set line = ReadLine of inFile  End while  Close inFile  Return omissionIDs | IDs to omit |

|  |  |  |
| --- | --- | --- |
| **CompareAllActivities** | | |
| **Input** | **Process** | **Output** |
| allActivities  omissionIDs | Processing Items:  None  Algorithm:  If length of omissionIDs NOT EQUAL to length of allActivities  Return 1  Else  Return -1  End if | Completion  Status flag |

|  |  |  |
| --- | --- | --- |
| **GetValidMenuChoice** | | |
| **Input** | **Process** | **Output** |
| menuChoiceIn  options  numOptions | Processing Items:  None  Algorithm:  Set menuChoice = 0  Input menuChoiceIn  Set goodInput = menuChoiceIn attempt parse as int to menuChoice  While goodInput IS NOT true OR (menuChoice < 1 OR  menuChoice > numOptions)  Call PromptValidOption of Prompts  Set i as an int = 0  While i < numOptions  Write options[i] to the console  Increment i by 1  End while  Input menuChoiceIn  Set goodInput = menuChoiceIn attempt parse as int to  menuChoice  end while  return menuChoice | Error-handled  Flow of  control |

|  |  |  |
| --- | --- | --- |
| **GetValidInt** | | |
| **Input** | **Process** | **Output** |
| number | Processing Items:  None  Algorithm:  Set userNum as an int = 0  Input number  Set goodInput = number attempt parse as int to userNum  While goodInput IS NOT true OR userNum < 1  Call PromptValidInt of Prompts  Input number  Set goodInput = number attempt parse as int to userNum  End while  Return userNum as a string | Error-handled  integer |

|  |  |  |
| --- | --- | --- |
| **ToArray** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Set optionsArray = a new string[], length GetCount of MenuDisplays  Set i as an int = 0  While i < GetCount of MenuDisplays  Set optionsArray[i] = GetOptions of menuOptions[i]  Increment i by 1  End while  Return optionsArray | Array of menu  options |

|  |  |  |
| --- | --- | --- |
| **GetRecommendations** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Call SetCount, pass 0, of Recommendations  Call SetCount, pass 0, of EventsCat  Call SetCount, pass 0, of FoodDrinksCat  Call SetCount, pass 0, of GamesCat  Call SetCount, pass 0, of MuseumsCat  Call SetCount, pass 0, of NatureCat  Call SetCount, pass 0, of NightLifeCat  Call SetCount, pass 0, of ShoppingCat  Call SetCount, pass 0, of ToursCat  Call SetCount, pass 0, of WorkshopsCat  Open completedFile, pass “completed.txt”  Set completeLine as a string = ReadLine of completedFile  If completeLine IS null  Call NoneCompleted of Prompts  Close completedFile  Return -1  End if  If completeLine IS null AND GetCount of Recommendations < 1  Call NoneRecommended of Prompts  Close completedFile  Return -1  End if  While completeLine IS NOT null  Set tempArray as a string[] = completeLine split by ‘#’  If tempArray[7] = “No” OR tempArray[7] = “Non” OR tempArray[7] =  “Nein”  Set completeLine = ReadLine of completedFile  Constinue to next iteration  Else  Open catFile, pass “MenuDisplays/CategoryMenu.txt”  Set catLine as a string = ReadLine of catFile  Set endCategoryLoop as a bool = false  Set spentNum as a string = tempArray[4], trim ‘$’ from start  While catLine IS NOT null AND endCategoryLoop IS false  Set catField as a string[] = catLine split by ‘#’  Set fieldSplit as a string[] = catField[2] split by “ “  If tempArray[8] = fieldSplit[1]  If catField[2] contains “1”  Set events[GetCount of EventsCat] = a new EventsCat, pass  tempArray[2], spentNum, tempArray[5]  Call IncCount of EventsCount  Else if catField[2] contains “2”  Set events[GetCount of FoodDrinkCat] = a new FoodDrinkCat, pass  tempArray[2], spentNum, tempArray[5]  Call IncCount of FoodDrinkCat  Else if catField[2] contains “3”  Set events[GetCount of GamesCat] = a new GamesCat, pass  tempArray[2], spentNum, tempArray[5]  Call IncCount of GamesCat  Else if catField[2] contains “4”  Set events[GetCount of MuseumsCat] = a new MuseumsCat, pass  tempArray[2], spentNum, tempArray[5]  Call IncCount of MuseumsCat  Else if catField[2] contains “5”  Set events[GetCount of NatureCat] = a new NatureCat, pass  tempArray[2], spentNum, tempArray[5]  Call IncCount of NatureCat  Else if catField[2] contains “6”  Set events[GetCount of NightLifeCat] = a new NightLifeCat, pass  tempArray[2], spentNum, tempArray[5]  Call IncCount of NightLifeCat  Else if catField[2] contains “7”  Set events[GetCount of ShoppingCat] = a new ShoppingCat, pass  tempArray[2], spentNum, tempArray[5]  Call IncCount of ShoppingCat  Else if catField[2] contains “8”  Set events[GetCount of ToursCat] = a new ToursCat, pass  tempArray[2], spentNum, tempArray[5]  Call IncCount of ToursCat  Else if catField[2] contains “9”  Set events[GetCount of WorkshopsCat] = a new WorkshopsCat,  pass tempArray[2], spentNum, tempArray[5]  Call IncCount of WorkshopsCat  Else  Call IncCount of Recommendations  Set endCategoryLoop = true  End if  Else  Set catLine = ReadLine of catFile  End if  End while  Set completeLine = ReadLine of completedFile  End if  End while  Close completedFile  Return 1 | Recommended  Activities |

|  |  |  |
| --- | --- | --- |
| **SortByPrice** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Set i as an int = 0  While i < GetCount -1 of EventsCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of EventsCat  If GetSpent of events[j] as an int > GetSpent of events[i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapEvents, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of FoodDrinkCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of FoodDrinkCat  If GetSpent of foodDrinks [j] as an int > GetSpent of  foodDrinks[i] as an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapFood, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of GamesCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of GamesCat  If GetSpent of games [j] as an int > GetSpent of games[i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapGames, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of MuseumsCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of MuseumsCat  If GetSpent of museums [j] as an int > GetSpent of museums[i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapMuseums, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of NatureCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of NatureCat  If GetSpent of natures [j] as an int > GetSpent of natures[i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapNatures, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of NightLifeCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of NightLifeCat  If GetSpent of nightLives [j] as an int > GetSpent of nightLives[i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapNightLives, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of ShoppingCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of ShoppingCat  If GetSpent of shoppings [j] as an int > GetSpent of shoppings[i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapShoppings, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of ToursCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of ToursCat  If GetSpent of tours [j] as an int > GetSpent of tours[i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapTours, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of WorkshopsCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of WorkshopsCat  If GetSpent of workshops[j] as an int > GetSpent of  workshops [i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapWorkshops, pass max, i  End if  Increment i by 1  End while | Sorted  Recommended  Activities |

|  |  |  |
| --- | --- | --- |
| **SwapEvents** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an EventsCat = events[x]  Set events[x] = events[y]  Set events[y] = temp | Swapped Events |

|  |  |  |
| --- | --- | --- |
| **SwapFood** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an FoodDrinksCat = foodDrinks[x]  Set foodDrinks [x] = foodDrinks [y]  Set foodDrinks [y] = temp | Swapped  FoodDrinks |

|  |  |  |
| --- | --- | --- |
| **SwapGames** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an GamesCat = games[x]  Set games [x] = games [y]  Set games [y] = temp | Swapped Games |

|  |  |  |
| --- | --- | --- |
| **SwapMuseums** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an MuseumsCat = museums[x]  Set museums [x] = museums [y]  Set museums [y] = temp | Swapped  Museums |

|  |  |  |
| --- | --- | --- |
| **SwapNatures** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an NatureCat = natures[x]  Set natures [x] = natures [y]  Set natures [y] = temp | Swapped  Natures |

|  |  |  |
| --- | --- | --- |
| **SwapNightLives** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an NightLifeCat = nightLives[x]  Set nightLives [x] = nightLives [y]  Set nightLives [y] = temp | Swapped  NightLives |

|  |  |  |
| --- | --- | --- |
| **SwapShoppings** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an ShoppingCat = shoppings[x]  Set shoppings [x] = shoppings [y]  Set shoppings [y] = temp | Swapped  Shoppings |

|  |  |  |
| --- | --- | --- |
| **SwapTours** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an ToursCat = tours[x]  Set tours [x] = tours [y]  Set tours [y] = temp | Swapped Tours |

|  |  |  |
| --- | --- | --- |
| **SwapWorkshops** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an WorkshopsCat = workshops[x]  Set workshops [x] = workshops [y]  Set workshops [y] = temp | Swapped  Workshops |

|  |  |  |
| --- | --- | --- |
| **CheckIfAnyActivities** | | |
| **Input** | **Process** | **Output** |
| remainActivities | Processing Items:  None  Algorithm:  If GetCount of Activity < 1  Call PromptNoActivities of Prompts  Call MainMenuPrompt of Prompts  Return -1  Else  Return 1  End if | Activity Remain  Flag |

|  |  |  |
| --- | --- | --- |
| **GetRemainingActivities** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Set allActivities as a string[,] = GetAllActivities  Set omissionIDs as a string[] = GetOmissionIDs  Set clearOmitIDs as a string[] = a new string, size GetCount of Activity  Call SetCount, pass 0, of RemainingActivities  If allActivities[0,0] = “-1”  Call NoneIncompleted of Prompts  Return -1  End if  If omissionIDs[0] = “-1”  Set i as an int = 0  While i < GetCount of Activity  Set minPrice as a string = allActivities[i,3] trim ‘$’ at start  Set maxPrice as a string = allActivities[i,4] trim ‘$’ at start  Set remainActivities[GetCount of RemainingActivities] =  A new RemainingActivities, pass allActivities[i,0], allActivities[i,1],  allActivities[i,2], minPrice, maxPrice, allActivities[i,5], allActivities[i,6]  Call IncCount of RemainingActivities  Increment i by 1  End while  Return 1  End if  If CompareAllActivities, pass allActivities, omissionIDs, < 0  Call NoneIncompleted of Prompts  Return -1  End if  Set i as an int = 0  While i < GetCount of Activity  Set j as an int = 0  While j < GetCount of CompleteActivity  If omissionIDs[j] = allActvities[i,0]  Break loop  Else if omissionIDs[j] NOT EQUAL to allActivities[i,0] AND  j NOT EQUAL to GetCount -1 of CompleteActivity  continue to next iteration  else  set remainActivities [GetCount of RemainingActivities] = a new  RemainingActivities, pass allActivities[i,0], allActivities[i,1]  Call IncCount of RemainingActivities  Set clearOmitIDs[i] = allActivities[i,0]  End if  Increment j by 1  End while  Increment i by 1  End while  Return 1 | Remaining  Activities |

|  |  |  |
| --- | --- | --- |
| **GetAllActivities** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Set count as an int = 0  Instantiate forComparision as an Activity[], length 50  Instantiate countActvities as an ActivitiyFile, pass forComparison  Call GetAllActivities of countActivities  Set allActivities as a string[,] = a new string, length [GetCount of Activity, 7]  Open inFile, pass “list.txt”  Set line as a string = ReadLine of inFile  If line IS null  Set noActivities as a string[,] = a new string, length [1, 7]  Set noActivities[0,0] = “-1”  Close inFile  Return noActivities  End if  While line IS NOT null  Set tempArray as a string[] = line split by ‘#’  Set i as an int = 0  While i < 7  Set allActivities[count, i] = tempArray[i]  Increment i by 1  End while  Increment count by 1  Set line = ReadLine of inFile  End while  Close inFile  Return allActivities | All activities |

|  |  |  |
| --- | --- | --- |
| **GetOmissionIDs** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Set count as an int = 0  Instantiate forComparision as a CompleteActivity[], length 50  Instantiate countCompletes as a CompleteActivityFile, pass forComparison  Call GetCompletedActivities of countCompletes  Set omissionIDs = a new string[], length GetCount of CompleteActivity  Open inFile, pass “completed.txt”  Set line as a string = ReadLine of inFile  If line IS null  Set noneCompleted as a string[] = {“-1”}  Close inFile  Return noneCompleted  End if  While line IS NOT null  Set tempArray as a string[] = line split by ‘#’  Set omissionIDs[count] = tempArray[1]  Increment count by 1  Set line = ReadLine of inFile  End while  Close inFile  Return omissionIDs | IDs to omit |

|  |  |  |
| --- | --- | --- |
| **CompareAllActivities** | | |
| **Input** | **Process** | **Output** |
| allActivities  omissionIDs | Processing Items:  None  Algorithm:  If length of omissionIDs NOT EQUAL to length of allActivities  Return 1  Else  Return -1  End if | Completion  Status flag |

|  |  |  |
| --- | --- | --- |
| **SortByCategory** | | |
| **Input** | **Process** | **Output** |
| sortChoice | Processing Items:  None  Algorithm:  Set i as an int = 0  While i < GetCount -1 of RemainingActivities  Set max as an int = i  Set j as an int = i+1  While j < GetCount of RemainingActivities  If sortChoice = 2  If CatCompareTo, pass remainActivities[j] of  remainActivities[max] > 0  set max = j  end if  else  if CatCompareTo, pass remainActivities[j] of  remainActivities[max] < 0  set max = j  end if  end if  increment j by 1  end while  if max IS NOT EQUAL to i  Call Swap, pass max, i  End if  End while | Sorted  Remaining  Activities |

|  |  |  |
| --- | --- | --- |
| **SortByTime** | | |
| **Input** | **Process** | **Output** |
| sortChoice | Processing Items:  None  Algorithm:  Set i as an int = 0  While i < GetCount -1 of RemainingActivities  Set max as an int = i  Set j as an int = i+1  While j < GetCount of RemainingActivities  Set amount as an int[] = ConvertTimeToMinutes, pass i, j  Set maxAmount as an int = amount[0]  Set nextAmount as an int = amount[1]  If sortChoice = 4  If nextAmount < maxAmount  Set max = j  End if  Else  If nextAmount > maxAmount  Set max = j  End if  End if  Increment j by 1  End while  If max is NOT EQUAL to i  Call Swap, pass max, i  End if  Increment i by 1  End while | Sorted  Remaining  Activities |

|  |  |  |
| --- | --- | --- |
| **ConvertTimeToMinutes** | | |
| **Input** | **Process** | **Output** |
| i  j | Processing Items:  None  Algorithm:  Set MINUTES\_IN\_A\_DAY as a constant int = 1440  Set MINUTES\_IN\_AN\_HOUR as a constant int = 60  Set maxAndMinTimes as an int[], length 2  Set tempArrayMax as a string[] = GetTimeNeeded of remainActivities[i]  Split by “-“  Set maxAmount as an int = tempArrayMax[0] as an int  Set maxCat as a string = tempArrayMax[1]  If maxCat = “Days” OR maxCat = “Day”  Set maxAmount = maxAmount X MINUTES\_IN\_AN\_HOUR  Else if maxCat = “Hours” OR maxCat = “Hour”  maxAmount = maxAmount X MINUTES\_IN\_AN\_HOUR  end if  set tempArrayNext as a string[] = GetTimeNeeded of remainActivities[j]  Split by “-“  Set nextAmount as an int = tempArrayNext[0] as an int  Set nextCat as a string = tempArrayNext[1]  If nextCat = “Days” OR nextCat = “Day”  Set nextAmount = nextAmount X MINUTES\_IN\_A\_DAY  Else if nextCat = “Hours” OR nextCat = “Hour”  Set nextAmount = nextAmount X MINUTES\_IN\_AN\_HOUR  End if  Set maxAndMinTimes[0] = maxAmount  Set maxAndMinTimes[1] = nextAmount  Return maxAndMinTimes | maxAndMinTimes |

|  |  |  |
| --- | --- | --- |
| **SortByPrice** | | |
| **Input** | **Process** | **Output** |
| sortChocie | Processing Items:  None  Algorithm:  Set i as an int = 0  While i < GetCount -1 of RemainingActivities  Set max as an int = i  Set j as an int = i+1  While j < GetCount of RemainingActivities  Declare a switch statement, variable sortChoice  In Case 6: if GetPriceMin of remainActivities[max] as an int >  GetPriceMin of remainActivities[j] as an int  Set max = j  Else if GetPriceMin of remainActivities[max] as an  Int = GetPriceMin of remainActivities[j] as an int  If GetPriceMax of remainActivities[max] as an int  > GetPriceMax of remainActivities[j] as an int  Set max = j  End if  End if  Break switch  In Case 7: if GetPriceMin of remainActivities[max] as an int <  GetPriceMin of remainActivities[j] as an int  Set max = j  Else if GetPriceMin of remainActivities[max] as an  Int = GetPriceMin of remainActivities[j] as an int  If GetPriceMax of remainActivities[max] as an int  < GetPriceMax of remainActivities[j] as an int  Set max = j  End if  End if  Break switch  In Case 8: if GetPriceMax of remainActivities[max] as an int >  GetPriceMax of remainActivities[j] as an int  Set max = j  Else if GetPriceMax of remainActivities[max] as an  Int = GetPriceMax of remainActivities[j] as an int  If GetPriceMin of remainActivities[max] as an int  > GetPriceMin of remainActivities[j] as an int  Set max = j  End if  End if  Break switch  Default to: if GetPriceMax of remainActivities[max] as an int <  GetPriceMax of remainActivities[j] as an int  Set max = j  Else if GetPriceMax of remainActivities[max] as an  Int = GetPriceMax of remainActivities[j] as an int  If GetPriceMin of remainActivities[max] as an int  < GetPriceMin of remainActivities[j] as an int  Set max = j  End if  End if  Break switch  End switch  Increment j by 1  End while  If max IS NOT EQUAL to i  Call Swap, pass max, i  End if  End while | Sorted  Remaining  Activities |

|  |  |  |
| --- | --- | --- |
| **SortByTicketNeeded** | | |
| **Input** | **Process** | **Output** |
| sortChoice | Processing Items:  None  Algorithm:  Set i as an int = 0  While i < GetCount -1 of RemainingActivities  Set max as an int = i  Set j as an int = i+1  While j , GetCount of RemainingActivities  If sortChoice = 7  If TicketCompareTo, pass remainActivities[j] of  remainActivities[max] > 0  set max = j  end if  else  If TicketCompareTo, pass remainActivities[j] of  remainActivities[max] < 0  set max = j  end if  end if  increment j by 1  end while  if max IS NOT EQUAL to i  Call Swap, pass max, i  End if  End while | Sorted  Remaining  Activities |

|  |  |  |
| --- | --- | --- |
| **Swap** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as a RemainingActivities = remainActivities[x]  Set remainActivities[x] = remainActivities[y]  Set remainActivities[y] = temp | Swapped  Remaining  Activities |

|  |  |  |
| --- | --- | --- |
| **DisplaySaveToFileOption** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Instantiate saveReportMenuDisplays as a Prompts[], length 1  Instantiate saveReportMenuOptions as a MenuDisplays[], length 2  Call SetMenuType, pass 14, of MenuDisplays  Instantiate saveReportMenu as a DisplaysFile, pass  saveReportMenuDisplays, saveReportMenuOptions  Call GetMenuDisplay of saveReportMenu  Instantiate displayCurrent as a DisplaysReport, pass  saveReportMenuDisplays, saveReportMenuOptions  Call DisplayText of displayCurrent  If GetSaveToFileChoice, pass saveReportMenuOptions = 2  Call MainMenuPrompt of Prompts  Return -1  Else  Clear the console  Return 1  End if | Save to file  Menu |

|  |  |  |
| --- | --- | --- |
| **GetSaveToFileChoice** | | |
| **Input** | **Process** | **Output** |
| saveReportMenuOptions | Processing Items:  None  Algorithm:  Instantiate convertOptions as a MenuDisplaysUtil, pass  saveReportMenuOptions  set options as a string[] = ToArray of convertOptions  Instantiate saveToFile as a Menu  Set saveToFile as a new Menu  Call SetNumOptions, pass length  of saveReportMenuOptions, of saveToFile  Call SetOptions, pass options, of saveToFile  Set menuChoice as an int = GetValidMenuChoice of  saveToFile  return menuChoice | menuChoice |

|  |  |  |
| --- | --- | --- |
| **GetExpenditures** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Call SetCount, pass 0, of Spending  Call SetCount, pass 0, of EventsCat  Call SetCount, pass 0, of FoodDrinkCat  Call SetCount, pass 0, of GamesCat  Call SetCount, pass 0, of MuseumsCat  Call SetCount, pass 0, of NatureCat  Call SetCount, pass 0, of NightLifeCat  Call SetCount, pass 0, of ShoppingCat  Call SetCount, pass 0, of ToursCat  Call SetCount, pass 0, of WorkshopsCat  Call SetTotSpentCat, pass “0”, of EventsCat  Call SetTotSpentCat, pass “0”, of FoodDrinkCat  Call SetTotSpentCat, pass “0”, of GamesCat  Call SetTotSpentCat, pass “0”, of MuseumsCat  Call SetTotSpentCat, pass “0”, of NatureCat  Call SetTotSpentCat, pass “0”, of NightLifeCat  Call SetTotSpentCat, pass “0”, of ShoppingCat  Call SetTotSpentCat, pass “0”, of ToursCat  Call SetTotSpentCat, pass “0”, of WorkshopsCat  Open completedFile, pass “completed.txt”  Set completeLine as a string = ReadLine of completedFile  If completeLine IS null  Call NoneCompleted of Prompts  Close completedFile  Return -1  End if  While completeLine IS NOT null  Set tempArray as a string[] = completeLine split by ‘#’  Set spent as an int = 0  Set amountSpent as a string = tempArray[4], trim ‘$’ at start  If amountSpent cannot be parsed into spent as an int  Set completeLine = ReadLine of completedFile  Constinue to next iteration  Else  Open catFile, pass “MenuDisplays/CategoryMenu.txt”  Set catLine as a string = ReadLine of catFile  Set endCategoryLoop as a bool = false  Set spentnum as a string = tempArray[4], trim ‘$’ at start  While catLine IS NOT null AND endCategoryLoop IS false  Set catField as a string[] = catLine split by ‘#’  Set fieldSplit as a string[] = catField[2] split by “ “  If tempArray[8] = fieldSplit[1]  If catField[2] contains “1”  Set events[GetCount of EventsCat] = a new EventsCat, pass  tempArray[2], tempArray[5]  Call SetTotSpentCat, pass (GetTotSpentCat of  EventsCat as an int + spentNum as an int) as a string,  Of EventsCat  Call IncCount of EventsCount  Else if catField[2] contains “2”  Set events[GetCount of FoodDrinkCat] = a new FoodDrinkCat, pass  tempArray[2], tempArray[5]  Call SetTotSpentCat, pass (GetTotSpentCat of  FoodDrinkCat as an int + spentNum as an int) as a string,  Of FoodDrinkCat  Call IncCount of FoodDrinkCat  Else if catField[2] contains “3”  Set events[GetCount of GamesCat] = a new GamesCat, pass  tempArray[2], tempArray[5]  Call SetTotSpentCat, pass (GetTotSpentCat of  GamesCat as an int + spentNum as an int) as a string,  Of GamesCat  Call IncCount of GamesCat  Else if catField[2] contains “4”  Set events[GetCount of MuseumsCat] = a new MuseumsCat, pass  tempArray[2], tempArray[5]  Call SetTotSpentCat, pass (GetTotSpentCat of  MuseumsCat as an int + spentNum as an int) as a string,  Of MuseumsCat  Call IncCount of MuseumsCat  Else if catField[2] contains “5”  Set events[GetCount of NatureCat] = a new NatureCat, pass  tempArray[2], tempArray[5]  Call SetTotSpentCat, pass (GetTotSpentCat of  NatureCat as an int + spentNum as an int) as a string,  Of NatureCat  Call IncCount of NatureCat  Else if catField[2] contains “6”  Set events[GetCount of NightLifeCat] = a new NightLifeCat, pass  tempArray[2], tempArray[5]  Call SetTotSpentCat, pass (GetTotSpentCat of  NightLifeCat as an int + spentNum as an int) as a string,  Of NightLifeCat  Call IncCount of NightLifeCat  Else if catField[2] contains “7”  Set events[GetCount of ShoppingCat] = a new ShoppingCat, pass  tempArray[2], tempArray[5]  Call SetTotSpentCat, pass (GetTotSpentCat of  ShoppingCat as an int + spentNum as an int) as a string,  Of ShoppingCat  Call IncCount of ShoppingCat  Else if catField[2] contains “8”  Set events[GetCount of ToursCat] = a new ToursCat, pass  tempArray[2], tempArray[5]  Call SetTotSpentCat, pass (GetTotSpentCat of  ToursCat as an int + spentNum as an int) as a string,  Of ToursCat  Call IncCount of ToursCat  Else if catField[2] contains “9”  Set events[GetCount of WorkshopsCat] = a new WorkshopsCat,  pass tempArray[2], tempArray[5]  Call SetTotSpentCat, pass (GetTotSpentCat of  WorkshopsCat as an int + spentNum as an int) as a string,  Of WorkshopsCat  Call IncCount of WorkshopsCat  Else  Call IncCount of Spending  Set endCategoryLoop = true  End if  Else  Set catLine = ReadLine of catFile  End if  End while  Set completeLine = ReadLine of completedFile  End if  End while  Close completedFile  Return 1 | Activity  Expendituires |

|  |  |  |
| --- | --- | --- |
| **SortByPrice** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Set i as an int = 0  While i < GetCount -1 of EventsCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of EventsCat  If GetSpent of events[j] as an int > GetSpent of events[i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapEvents, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of FoodDrinkCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of FoodDrinkCat  If GetSpent of foodDrinks [j] as an int > GetSpent of  foodDrinks[i] as an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapFood, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of GamesCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of GamesCat  If GetSpent of games [j] as an int > GetSpent of games[i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapGames, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of MuseumsCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of MuseumsCat  If GetSpent of museums [j] as an int > GetSpent of museums[i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapMuseums, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of NatureCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of NatureCat  If GetSpent of natures [j] as an int > GetSpent of natures[i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapNatures, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of NightLifeCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of NightLifeCat  If GetSpent of nightLives [j] as an int > GetSpent of nightLives[i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapNightLives, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of ShoppingCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of ShoppingCat  If GetSpent of shoppings [j] as an int > GetSpent of shoppings[i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapShoppings, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of ToursCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of ToursCat  If GetSpent of tours [j] as an int > GetSpent of tours[i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapTours, pass max, i  End if  Increment i by 1  End while  Set i as an int = 0  While i < GetCount -1 of WorkshopsCat  Set max as an int = i  Set j as an int = i+1  While j < GetCount of WorkshopsCat  If GetSpent of workshops[j] as an int > GetSpent of  workshops [i]  As an int  Set max = j  End if  Increment j by 1  End while  If max IS NOT EQUAL to i  Call SwapWorkshops, pass max, i  End if  Increment i by 1  End while | Sorted  Completed  Activities |

|  |  |  |
| --- | --- | --- |
| **SwapEvents** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an EventsCat = events[x]  Set events[x] = events[y]  Set events[y] = temp | Swapped Events |

|  |  |  |
| --- | --- | --- |
| **SwapFood** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an FoodDrinksCat = foodDrinks[x]  Set foodDrinks [x] = foodDrinks [y]  Set foodDrinks [y] = temp | Swapped  FoodDrinks |

|  |  |  |
| --- | --- | --- |
| **SwapGames** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an GamesCat = games[x]  Set games [x] = games [y]  Set games [y] = temp | Swapped Games |

|  |  |  |
| --- | --- | --- |
| **SwapMuseums** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an MuseumsCat = museums[x]  Set museums [x] = museums [y]  Set museums [y] = temp | Swapped  Museums |

|  |  |  |
| --- | --- | --- |
| **SwapNatures** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an NatureCat = natures[x]  Set natures [x] = natures [y]  Set natures [y] = temp | Swapped  Natures |

|  |  |  |
| --- | --- | --- |
| **SwapNightLives** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an NightLifeCat = nightLives[x]  Set nightLives [x] = nightLives [y]  Set nightLives [y] = temp | Swapped  NightLives |

|  |  |  |
| --- | --- | --- |
| **SwapShoppings** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an ShoppingCat = shoppings[x]  Set shoppings [x] = shoppings [y]  Set shoppings [y] = temp | Swapped  Shoppings |

|  |  |  |
| --- | --- | --- |
| **SwapTours** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an ToursCat = tours[x]  Set tours [x] = tours [y]  Set tours [y] = temp | Swapped Tours |

|  |  |  |
| --- | --- | --- |
| **SwapWorkshops** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an WorkshopsCat = workshops[x]  Set workshops [x] = workshops [y]  Set workshops [y] = temp | Swapped  Workshops |

|  |  |  |
| --- | --- | --- |
| **GetDayByDayCompletions** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Call SetCount, pass 0, of TripSummaryByDay  Open completedFile, pass “completed.txt”  Set completeLine as a string = ReadLine of completedFile  If completedLine IS null  Call NoneCompleted of Prompts  Close completedFile  Return -1  End if  While completeLine IS NOT null  Set tempArray as a string[] = completeLine split by ’#’  Set dayByDayCompletions[GetCount of TripSummaryByDay] = a  TripSummaryByDay, pass tempArray[0], tempArray[1], tempArray[2],  tempArray[3]  set completeLine = ReadLine of completedFIle  end while  close completedFile  return 1 | Completed  Activities |

|  |  |  |
| --- | --- | --- |
| **SortByDay** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Set i as an int = 0  While i < GetCount -1 of TripSummaryByDay  Set max as an int = i  Set j as an int = i+1  While j < GetCount of TripSummaryByDay  Set maxDateArray as a string[] = GetDateComplete of  dayByDayCompletions[max] split by “/”  Set nextDateArray as a string[] = GetDateComplete of  dayByDayCompletions[j] split by “/”  if nextDateArray[2] as an int >= maxDateArray[2] as an int  if nextDateArray[0] as an int >= maxDateArray[0] as an int  if nextDateArray[1] as an int > maxDateArray[1] as an int  set max = j  else if nextDateArray[1] as an int <= maxDateArray[1] as an int AND  nextDateArray[0] as an int > maxDateArray[0] as an int  max = j  end if  else if nextDateArray[0] as an int < maxDateArray[0] as an int AND  nextDateArray[2] as an int > maxDateArray[2] as an int  max = j  end if  end if  increment j by 1  end while  if max IS NOT EQUAL to i  Call Swap, pass max, i  End if  Increment i by 1  End while | Completed  Activities  Sorted by day |

|  |  |  |
| --- | --- | --- |
| **Swap** | | |
| **Input** | **Process** | **Output** |
| X  Y | Processing Items:  None  Algorithm:  Set temp as an TripSummaryByDay = dayByDayCompletions[x]  Set dayByDayCompletions [x] = dayByDayCompletions [y]  Set dayByDayCompletions [y] = temp | Swapped  Completed  Activities |

|  |  |  |
| --- | --- | --- |
| **GetAllVacations** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Call SetCount, pass 0, of Vacation  Open inFile, pass ”vacation.txt”  Set line as a string = ReadLine of inFile  While line IS NOT null  Set tempArray as a string[] = line split by ‘#’  Set vacations[GetCount of Vacation] = a Vacation, pass tempArray[0],  tempArray[1], tempArray[2], tempArray[3], tempArray[4]  Call IncCount of Vacation  Set line = ReadLine of inFile  End while  Close inFile | Vacations |

|  |  |  |
| --- | --- | --- |
| **GetNewVacationInfo** | | |
| **Input** | **Process** | **Output** |
| vacations | Processing Items:  None  Algorithm:  Instantiate newVacationTool as a VacationUtil  Clear the console  Set vacations[GetCount of Vacation] = a new Vacation  Call SetID, pass GenNewID, of vacations[GetCount of Vacation]  Call PromptNewDestination of Prompts  Call SetDestination, pass GetValidDestination, pass input, of  newVacationTool, of vacations[GetCount of Vacation]  Call PromptNewStartDate of Prompts  Set dateType as an int = -1  Call SetStartDate, pass GetValidDate, pass dateType, vacations, of  newVacationTool, of vacations[GetCount of Vacation]  Call PromptNewEndDate of Prompts  Set dateType = 1  Call SetEndDate, pass GetValidDate, pass dateTypem vacations, of  newvacationsTool, of vacations[GetCount of Vacation]  Call PromptNewBudget of Prompts  Call SetBudget, pass GetValidBudget, pass input, of newVacationTool,  Of vacations[GetCount of Vacation]  Call NewVacationToFile, pass vacations  Call IncCount of Vacation  Call NewVacationSuccess of Prompts | New vacation  details |

|  |  |  |
| --- | --- | --- |
| **GenNewID** | | |
| **Input** | **Process** | **Output** |
| None | Processing Items:  None  Algorithm:  Open inFile, pass “vacation.txt”  Set line as a string = ReadLine of inFile  If line IS null  Close inFile  Return “1”  End if  Set checkVacationIDs as a new string[], length GetCount+1 of Vacation  Set index as an int = 0  While line IS NOT null  Set tempArray as a string[] = line split by ‘#’  Set checVacationIDs[index] = tempArray[0]  Increment index by 1  Set line = ReadLine of inFile  End while  Close inFile  Set maxID as an int = 0  Set i as an int = 0  While i < GetCount of Vacation  If checkvacationIDs[i] as an int > maxID  Set maxID = checkVacationIDs[i] as an int  End if  Increment i by 1  End while  Return maxID+1 as a string | New Vacation ID |

|  |  |  |
| --- | --- | --- |
| **CheckID** | | |
| **Input** | **Process** | **Output** |
| trueInt | Processing Items:  None  Algorithm:  Set count as an int = 0  Set goodID as a bool = false  While goodID IS NOT true AND trueInt IS NOT -1  Open inFile, pass “vacation.txt”  Set line as a string = ReadLine of inFile  Set count = 0  While line IS NOT null  Set tempArray as a string[] = line split by ‘#’  If trueInt IS NOT EQUAL to tempArray[0] as an int  Increment count by 1  Set line = ReadLine of inFile  Continue to next iteration  Else  Set goodID = true  Break loop  End if  End while  If line IS null  Do  Call PromptValidOrAbort of Prompts  While input cannot by parsed as an int to trueInt AND trueInt IS NOT  -1  End if  Close inFile  End while  If trueInt IS NOT -1  Return count  Else  Return trueInt  End if | New Vacation ID |

|  |  |  |
| --- | --- | --- |
| **GetUserLineToEdit** | | |
| **Input** | **Process** | **Output** |
| vacEditID | Processing Items:  None  Algorithm:  Set trueInt as an int = 0  Set verifiedInt as a string = VerifyEditInt, pass vacEditID  If verifiedInt as lower case = “stop”  Return -1  Else  Set trueInt = verifiedInt as an int  End if  Set lineToEdit as an int = CheckID, pass trueInt  Return lineToEdit | lineToEdit |

|  |  |  |
| --- | --- | --- |
| **VerifyEditInt** | | |
| **Input** | **Process** | **Output** |
| vacEditID | Processing Items:  None  Algorithm:  Set userID as an int = 0  Set goodInput as a bool = vacEditID attempt parse as int to userID  While goodInput IS NOT true AND vacEditID as lower case NOT “stop”  Call PromptvalidEditID of Prompts  Input vacEditID  Set goodInput = vacEditID attempt parse as int to userID  End while  Return vacEditID | vacEditID |

|  |  |  |
| --- | --- | --- |
| **NewVacationToFile** | | |
| **Input** | **Process** | **Output** |
| vacations | Processing Items:  None  Algorithm:  Open outFile, pass “vacation.txt”, in append mode  Write GetConcatVacation of vacations[GetCount of Vacation] to outFile  Close outFile | Vacation  Output  To File |

|  |  |  |
| --- | --- | --- |
| **EditVacationToFile** | | |
| **Input** | **Process** | **Output** |
| vacations | Processing Items:  None  Algorithm:  Open outFile, pass “vacation.txt”, in append mode  Set i as an int = 0  While i < GetCount of Vacation  Write GetConcatVacation of vacations[i] to outFile  Increment i by 1  End while  Close outFile | Vacation  Output  To File |

|  |  |  |
| --- | --- | --- |
| **DeleteVacationFromFile** | | |
| **Input** | **Process** | **Output** |
| lineToEdit  vacations | Processing Items:  None  Algorithm:  Open outFile, pass “vacation.txt”  Set i as an int = 0  While i < GetCount of Vacation  If i = lineToEdit  Continue to next iteration  Else  Write GetConcatVacation of vacations[i] to outFile  End if  Increment i by 1  End while  Close outFile | Vacation  Output  To File |

|  |  |  |
| --- | --- | --- |
| **EditVacationField** | | |
| **Input** | **Process** | **Output** |
| fieldEditChoice  lineToEdit | Processing Items:  None  Algorithm:  Set tempVacCount as an int = GetCount of Vacation  Call SetCount, pass lineToEdit, of Vacation  Declare a switch statement, variable fieldEditChoice  In Case 1: Call PromptDestination of Prompts  Input newField  If newField as lower case = “stop”  Break switch  Else  Call SetDestination, pass newField, of  vacations[lineToEdit]  Break switch  In Case 1: Call PromptStartDate of Prompts  Set newField = GetValidDate, pass -1, vacations  If newField as lower case = “stop”  Break switch  Else  Call SetStartDate, pass newField, of  vacations[lineToEdit]  Break switch  In Case 1: Call PromptEndDate of Prompts  Set newField = GetValidDate, pass 1, vacations  If newField as lower case = “stop”  Break switch  Else  Call SetEndDate, pass newField, of  vacations[lineToEdit]  Break switch  In Case 1: Call PromptBudget of Prompts  Input newField  If newField as lower case = “stop”  Break switch  Else  Call SetBudget, pass GetValidBudget, pass, newField, of  vacations[lineToEdit]  Break switch  Default to: break switch  End switch  Call SetCount, pass tempVacCount, of Vacation | Updated  Vacation  Details |

|  |  |  |
| --- | --- | --- |
| **GetValidDestination** | | |
| **Input** | **Process** | **Output** |
| destination | Processing Items:  None  Algorithm:  If length of destination > 0  Return destination  Else  Return GetPlaceHolderDestination of Prompts  End if | Valid  Destination |

|  |  |  |
| --- | --- | --- |
| **GetValidBudget** | | |
| **Input** | **Process** | **Output** |
| budget | Processing Items:  None  Algorithm:  Set foodBudget as an int = 0  Set goodInput as a bool = budget attempt parse as int to goodBudget  While goodInput IS NOT true OR goodBudget < 0  Call PromptValidBudget of Prompts  Input budget  Set goodInput = budget attempt parse as int to goodBudget  End while  Return budget | Valid  Budget |

|  |  |  |
| --- | --- | --- |
| **GetValidDate** | | |
| **Input** | **Process** | **Output** |
| dateType  vacations  userDate | Processing Items:  None  Algorithm:  Input userDate  Set dateArray as a string[] = userDate split by “/”  Set monthSpecifier as an int = 0  Set daysInMonths as an int = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}  Instantiate dateError as a VacationReport  If userDate as lower case = “stop  Return userDate  End if  Set i as an int = 0  While i < 3  Set failedCheck as a bool = false  Set number as an int = 0  Set notADay as a bool = false  Set userNumber as a string = dateArray[i]  Set goodInput as a bool = userNumber attempt parse as int to number  If i = 0  Set monthSpecifier = number  End if  If i = 1  If number < 1 OR number > daysInMonths[monthSpecifier -1 ]  Set notADay = true  End if  End if  If goodInput IS NOT true OR length of dateArray < 3 OR length of dateArray  > 3 OR (i = 0 AND (number < 1 OR number > 12)) OR notADay IS true  OR (i = 2 AND (number < 1999 OR number > 2099))  Call PromptDateError, pass 4, -1, of Prompts  Set failedCheck = true  Set i = -1  End if  Set endYear as an int = dateArray[2] as an int  If (i > -1 AND i < 3) AND dateType = 1  Set startDate as a string[] = GetStartDate of  vacations[GetCount of Vacation] split by “/”  if i = 0 AND number < startDate[0] as an int AND endYear <= dartDate[2]  as an int  Call PromptDateError, pass i+1, dateType, of Prompts  Set failedCheck = true  Set i = -1  End if  If i = 1 AND number < startDate[1] as an int AND monthSpecifier <=  startDate[0] as an int AND endYear <= startDate[2] as an int  Call PromptDateError, pass i+1, dateType, of Prompts  Set failedCheck = true  Set i = -1  End if  If i = 2  Set year as an int = 0  Set yearCheck as a bool = startDate[2] attempt parse as int to year  If yearCheck IS NOT true OR number < year  Call PromptDateError, pass i+1, dateType, of Prompts  Set failedCheck = true  Set i = -1  End if  End if  End if  If failedCheck = true  Input userDate  Set dateArray = userDate split by “/”  End if  Increment i by 1  End while  Return userDate | userDate |

|  |  |  |
| --- | --- | --- |
| **AssignCatLanguage** | | |
| **Input** | **Process** | **Output** |
| catChoice | Processing Items:  None  Algorithm:  Open inFile, pass “MenuDisplays/CategoryMenu.txt”  Set line as a string = ReadLine of inFIle  While line IS NOT null  Set languageArray as a string[] = line split by ‘#’  If languageArray[0] as an int = GetLanguageID of Prompts  Set line = ReadLine of inFile  Set i as an int = 1  While i < 10  If i = catChoice  Set lineArray as a string[] = line split by ‘#’  Set categoryField as a string[] = lineArray[2] split by ‘ ‘  Return categoryField[1]  Else  Set line = ReadLine of inFile  End if  Increment i by 1  End while  End if  Set line = ReadLine of inFile  End while  Close inFile  Return “CAT\_NOT\_FOUND” | System-requested  category in the  current language |

|  |  |  |
| --- | --- | --- |
| **AssignRating** | | |
| **Input** | **Process** | **Output** |
| ratingChoice | Processing Items:  None  Algorithm:  Open inFile, pass “MenuDisplays/ProcessRating.txt”  Set line as a string = ReadLine of inFIle  While line IS NOT null  Set languageArray as a string[] = line split by ‘#’  If languageArray[0] as an int = GetLanguageID of Prompts  Set line = ReadLine of inFile  Set i as an int = 1  While i < 10  If i = ratingChoice  Set lineArray as a string[] = line split by ‘#’  Set ratingField as a string[] = lineArray[2] split by ‘ ‘  Return ratingField[1] + “ “ + ratingField[2]  Else  Set line = ReadLine of inFile  End if  Increment i by 1  End while  End if  Set line = ReadLine of inFile  End while  Close inFile  Return “RATING\_NOT\_FOUND” | System-requested  rating in the  current language |

|  |  |  |
| --- | --- | --- |
| **AssignRating** | | |
| **Input** | **Process** | **Output** |
| recommend  Choice | Processing Items:  None  Algorithm:  Open inFile, pass “MenuDisplays/ProcessRecommend.txt”  Set line as a string = ReadLine of inFIle  While line IS NOT null  Set languageArray as a string[] = line split by ‘#’  If languageArray[0] as an int = GetLanguageID of Prompts  Set line = ReadLine of inFile  Set i as an int = 1  While i < 10  If i = recommendChoice  Set lineArray as a string[] = line split by ‘#’  Set recommendField as a string[] = lineArray[2] split by ‘ ‘  Return recommendField[1]  Else  Set line = ReadLine of inFile  End if  Increment i by 1  End while  End if  Set line = ReadLine of inFile  End while  Close inFile  Return “RECOMMENDATION\_NOT\_FOUND” | System-requested  recommendation  in the  current language |

|  |  |  |
| --- | --- | --- |
| **GetAllFiles** | | |
| **Input** | **Process** | **Output** |
| allFiles | Processing Items:  None  Algorithm:  Open inFile, pass “FileVerification/textAssets.txt”  Set line as a string = ReadLine of inFile  Set count as an int = 0  While line IS NOT null  Set allFiles[count] = line  Increment count by 1  Set line = ReadLine of inFile  End while  Close inFile | allFiles |

|  |  |  |
| --- | --- | --- |
| **GetMissingFiles** | | |
| **Input** | **Process** | **Output** |
| allFiles  missingFiles  NUM\_FILES | Processing Items:  None  Algorithm:  Set i as an int = 0  While i < NUM\_FILES  If allFiles[i] DOES NOT exist  Set missingFiles[i] = allFiles[i]  End if  Increment i by 1  End while | missingFiles |

|  |  |  |
| --- | --- | --- |
| **DisplayMissingFiles** | | |
| **Input** | **Process** | **Output** |
| allFiles  missingFiles | Processing Items:  None  Algorithm:  Set NUM\_FILES as a constant int = 38  Set allFiles as a new string[], length NUM\_FILES  Set missingFiles as a new string[], length NUM\_FILES  Call GetAllFiles, pass allFiles by reference  Call GetMissingFiles, pass allFiles, missingFiles by reference, NUM\_FILES  Set missingAsset as a bool = false  Set i as an int = 0  While i < length of missingFiles  If missingFiles[i] IS NOT null  Set missingAsset = true  Break loop  End if  Increment i by 1  End while  If missingAsset = false  Return  Else  Call PromptMissingFiles of Prompts  Set i as an int = 0  While i < length of missingFiles  If missingFiles[i] IS NOT null  Write missingFiles[i] and a new line to the console  End if  Increment i by 1  End while  Call PromptRestoreFiles of Prompts  Exit the program  End if | Missing files |