# AQA Text Adventures

**Programming Tasks (C#)**

**The following require you to open the skeleton program and make modifications**

## **Task 1**

This question refers to PlayGame as well as a new subroutine ShowHelp.

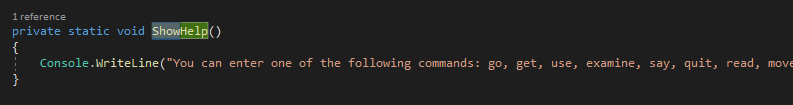
Currently, users are not offered any help regarding valid commands. If a user enters an invalid command, they are told ‘Sorry, you don’t know how to…’. Add a new subroutine called ShowHelp, which should display the following text on two lines:

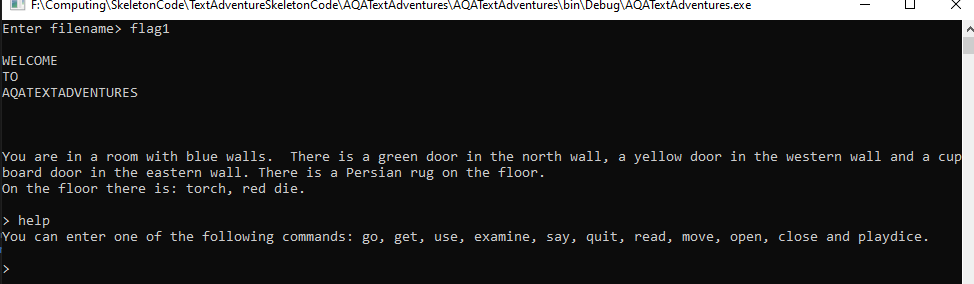
You can enter one of the following commands:

go, get, use, examine, say, quit, read, move, open, close and playdice.

Modify the subroutine PlayGame so that if a user enters the command ‘help’, the subroutine ShowHelp is called.

|  |
| --- |
| **Evidence you need to provide:**   * Your amended SOURCE CODE for PlayGame * Your SOURCE CODE for the new ShowHelp subroutine * One screenshot showing the output that results from the following:   + Run the program and load the file ‘flag1’   + Type ‘help’ at the first prompt |





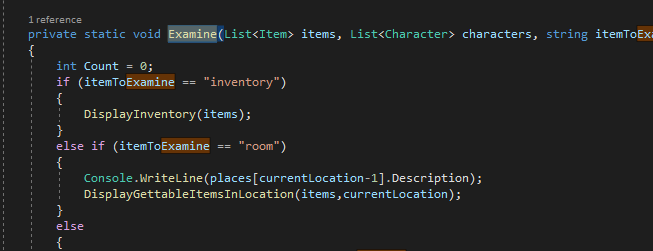
## **Task 2**

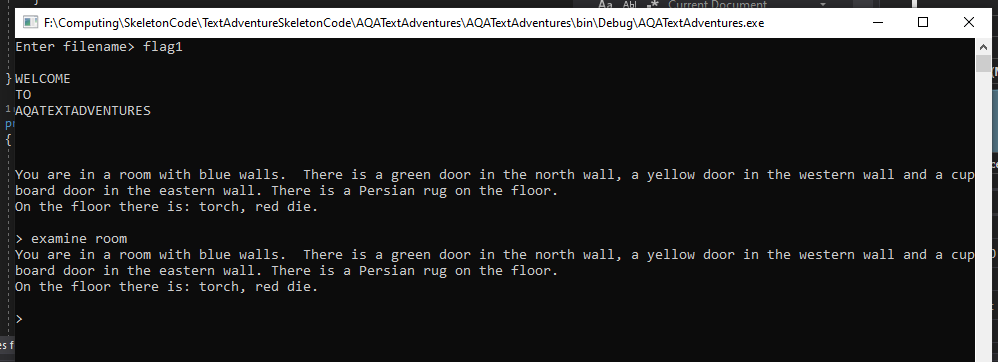
This question refers to Examine and PlayGame.

Currently, users are able to examine their inventory, as well as objects within a room, but not the room itself. Modify Examine so that if the user has entered the text ‘examine room’, the description of the current location will be re-displayed. A call to DisplayGettableItemsInLocation should also be made.

You should pass the places List from PlayGame to Examine.

|  |
| --- |
| **Evidence you need to provide:**   * Your amended SOURCE CODE for Examine * Your amended SOURCE CODE for PlayGame * One screenshot showing the output that results from the following:   + Run the program and load the file ‘flag1’   + Type ‘examine room’ at the first prompt |



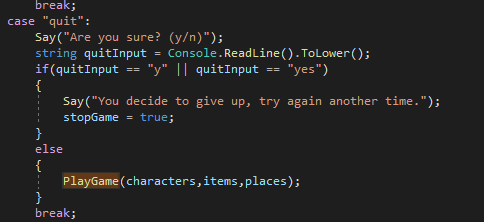


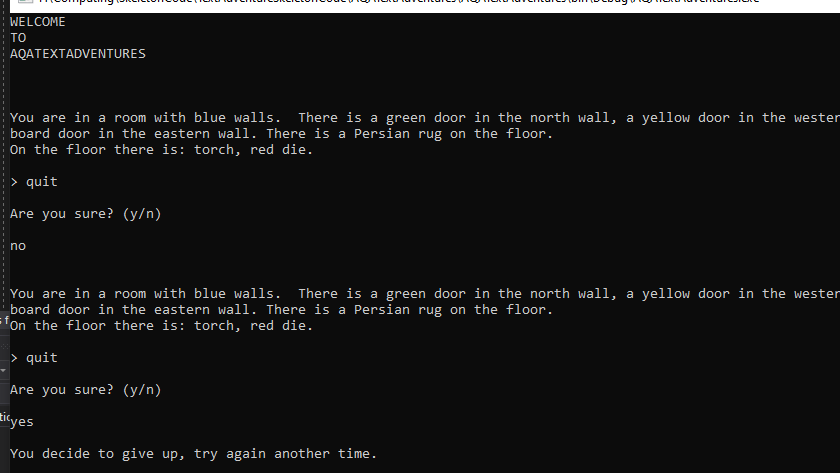
## **Task 3**

This question refers to PlayGame.

Currently, if the user types ‘quit’ at the prompt, the game ends without any further input from them. Modify PlayGame so that an entry of ‘quit’ triggers an output of ‘Are you sure? (y/n)’ and an additional prompt. Entering either ‘y’ or ‘yes’ in any combination of upper or lowercase should terminate the game as previously, with ‘You decide to give up, try again another time’ still being displayed. Any other input should return the player to the game and the main prompt.

|  |
| --- |
| **Evidence you need to provide:**   * Your amended SOURCE CODE for PlayGame * One screenshot showing the output that results from the following:   + Run the program and load the file ‘flag1’   + Type ‘quit’ at the first prompt   + Type ‘no’ in response to ‘Are you sure’   + Type ‘quit’ at the next prompt   + Type ‘yes’ in response to ‘Are you sure’   + Press enter after resulting output |





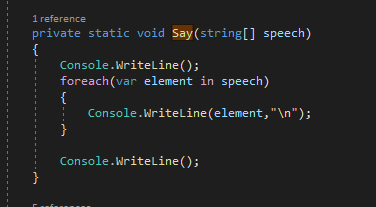
## **Task 4**

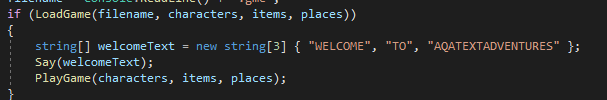
This question refers to Say and Main.

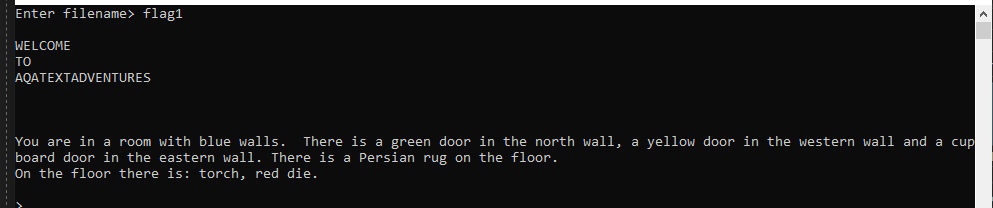
Currently, output that is displayed as a result of a call to Say is displayed in a single line. For multiple lines to be displayed, multiple calls to Say must be made. Create an additional implementation of Say that receives a string array instead of a string. It should then output the content of each element of the array on separate lines, maintaining the line breaks both before and after the output.

In order to test your modified version of Say, you will need to add code to the Main subroutine. Immediately before the call to PlayGame, create an array consisting of the strings “WELCOME”, “TO” and “AQATEXTADVENTURES” and pass this array to Say.

|  |
| --- |
| **Evidence you need to provide:**   * Your amended SOURCE CODE for Say * Your amended SOURCE CODE for Main * One screenshot showing the output that results from the following:   + Run the program and load the file ‘flag1’ |





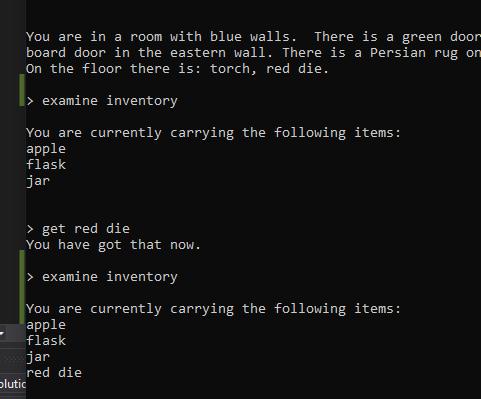


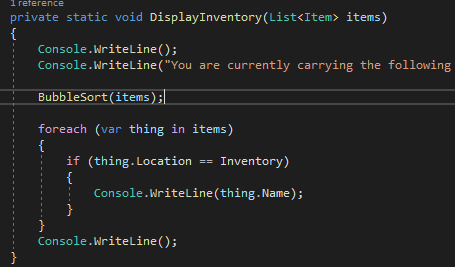
## **Task 5**

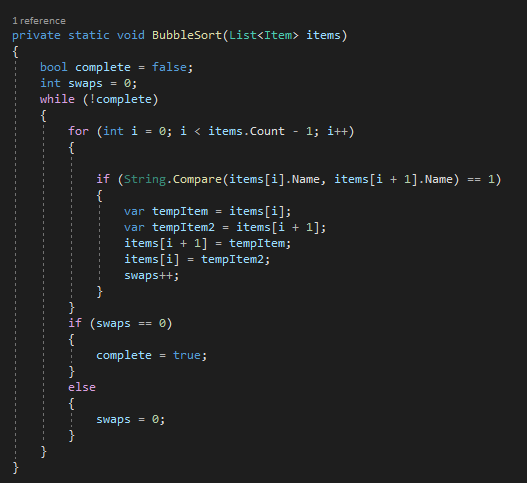
This question refers to DisplayInventory.

Currently, when the inventory is examined items are displayed in the order in which they were loaded from the binary file. Modify the DisplayInventory subroutine, so that it uses a bubble sort to sort the items into ascending alphabetical order based on their Name attribute before displaying the list of items. You should devise the sort algorithm yourself, and should not use any library sorting functions.

|  |
| --- |
| **Evidence you need to provide:**   * Your amended SOURCE CODE for DisplayInventory * One screenshot showing the output that results from the following:   + Run the program and load the file ‘flag1’   + Type ‘examine inventory’ at the first prompt   + Type ‘get red die’ at the second prompt   + Type ‘examine inventory’ at the third prompt |







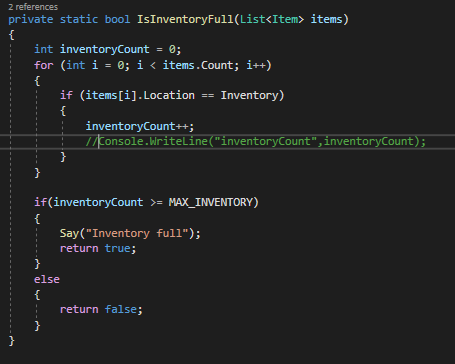
## **Task 6**

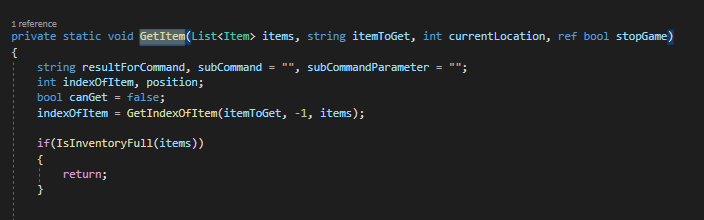
This question refers to GetItem and TakeItemFromOtherCharacter as well as the constants.

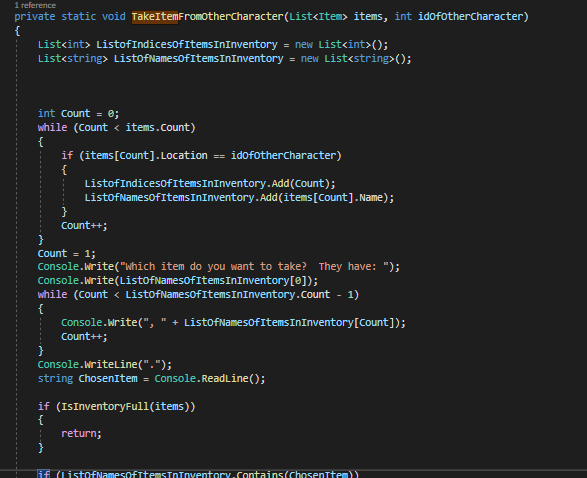
Currently, the player’s inventory can contain as many items as can be found. Create an integer constant called MAX\_INVENTORY and set it to 4.

When an attempt is made to get an item by a player with four or more items already in the inventory, the message ‘Inventory full’ should be displayed. The check for a full inventory should take place before any existing selection statements. If the inventory is full, no messages should be output from within GetItem other than ‘Inventory full’. This check should also be applied to TakeItemFromOtherCharacter so that when the player selects an item after winning a dice game, the message ‘Inventory full’ will be displayed if the player already has a full inventory.

|  |
| --- |
| **Evidence you need to provide:**   * Your SOURCE CODE for the new MAX\_INVENTORY constant * Your amended SOURCE CODE for GetItem * One screenshot showing the output that results from the following:   + Run the program and load the file ‘flag1’   + Type ‘get torch’ at the first subsequent prompt   + Type ‘get red die’ at the second prompt   + Type ‘examine inventory’ at the third prompt * Your amended SOURCE CODE for TakeItemFromOtherCharacter * One screenshot showing the output that results from the following:   + Run the program and load the file ‘flag2’   + Type ‘playdice guard’ at the first subsequent prompt   + Type ‘6’ and ‘6’ for the minimum and maximum at the second and third prompts   + Repeat the previous two steps if the result is a draw   + Type ‘jar’ if you win at the final prompt |







## **Task 7**

This question refers to PlayGame as well as a new subroutine DropItem.

Currently, items can be picked up but not dropped. Create a new subroutine called DropItem which accepts three parameters:

* A List called ‘items’, which will contain all items in the game
* A string called ‘itemToDrop’, which will be the name of the item leaving the inventory
* An integer called ‘currentLocation’, which will be the ID of the player’s location

DropItem should check that the item exists and that the player is carrying the item. If either of these is not *true*, the text ‘You are not carrying a \_\_\_\_\_’ should be displayed, with the item name displayed in place of \_\_\_\_\_. Otherwise, the item should be placed into the room with the text ‘\_\_\_\_\_ dropped’ indicating which item was dropped.

You should modify PlayGame to call DropItem if a command of ‘drop’ is entered by the player.

|  |
| --- |
| **Evidence you need to provide:**   * Your amended SOURCE CODE for PlayGame * Your SOURCE CODE for the new DropItem subroutine * One screenshot showing the output that results from the following:   + Run the program and load the file ‘flag1’   + Type ‘drop flask’ at the first prompt   + Type ‘drop flask’ again at the second prompt   + Type ‘examine inventory’ at the third prompt |

## 

## **Task 8**

This question refers to Main.

Currently, if the user enters the name of a non-existent file, the program ends without allowing the user to enter a different file name. Additionally, the file extension .gme is always appended to the file name, even if the user has included the extension themselves. This would result in a file not being found.

Modify the user prompt in Main to read ‘Enter filename or enter Q to quit’. If the user enters ‘Q’, the program should end. If the user enters a valid file name, the file should load as normal. Otherwise, the prompt should continually re-display until the user has entered ‘Q’ or a file name that successfully loads a file. If the user enters a file name that contains the string .gme the program should not append .gme to their file name.

You should only make changes to Main.

|  |
| --- |
| **Evidence you need to provide:**   * Your amended SOURCE CODE for Main * One screenshot showing the output that results from the following:   + Type ‘flag3’ when prompted for a file name   + At the next prompt, type ‘flag1.gme’ |

## **Task 9**

This question relates to PlayGame as well as a new subroutine DisplayCharacters.

In order for a developer to work effectively on existing code, it is helpful for them to be able to quickly view the content of a data structure. Create a new subroutine called DisplayCharacters, which should accept a single parameter in the form of the characters List.

When DisplayCharacters is called, it should display four column headers in this order:

* ID (followed by three spaces)
* NAME (followed by eight spaces)
* DESCRIPTION (followed by 53 spaces)
* CURRENTLOCATION

Beneath these columns, all characters within the game should be displayed, with each on a new line. Their values for the attributes ID, Name, Description and CurrentLocation should each line up beneath the relevant column header.

Modify PlayGame so that if ‘debugchar’ is entered during gameplay, a call to DisplayCharacters will be made.

|  |
| --- |
| **Evidence you need to provide:**   * Your amended SOURCE CODE for PlayGame * Your SOURCE CODE for the new DisplayCharacters subroutine * One screenshot showing the output that results from the following:   + Type ‘flag1’ when prompted for a file name   + At the next prompt, type ‘debugchar’ |

## **Task 10**

This question relates to PlayGame as well as a new subroutine FillVessel.

In the flag1 game, there is a room below the starting location that contains a barrel of water. Create a new subroutine called FillVessel that requires the following parameters

* characters List
* items Lists
* places List
* a String called ‘vessel’

FillVessel should check whether the player and the barrel are in the same location and that the vessel is an item whose Status attribute contains the text ‘container’ and does not contain the text ‘large’. The vessel should also be in the player’s inventory. If any of these conditions is not met, the message ‘You cannot do that’ should be displayed. Otherwise the vessel item’s Name attribute should have ‘ of water’ appended to it, and the text ‘You have filled the ’ should be displayed, with the vessel appended to the end. If the vessel’s name attribute already contains the text ‘ of water’, the text ‘Already full’ should be displayed.

Add a call to FillVessel in PlayGame that will occur if the player enters the command ‘fill’.

|  |
| --- |
| **Evidence you need to provide:**   * Your amended SOURCE CODE for PlayGame * Your SOURCE CODE for the new FillVessel subroutine * One screenshot showing the output that results from the following:   + Type ‘flag1’ when prompted for a file name   + At the next prompt, type ‘open trapdoor’   + At the next prompt, type ‘go down’   + At the next prompt, type ‘fill flask’   + At the next prompt, type ‘fill apple’   + At the next prompt, type ‘fill flask of water’   + At the next prompt, type ‘examine inventory’ |

## **Task 11**

This question relates to Go, PlayGame and a new class, History.

Create a new class called History, with two attributes:

* An integer called pointer, initially set to -1
* A five-element integer array called locations

History will use a stack, to allow players to backtrack through previous locations by typing ‘go back’. The stack will use an implementation of the locations array, which will be subject to two subroutines:

* Push should require an integer as a parameter. If the stack is full, all existing content should be moved down by 1 element (so the content of element 0 is lost), and the value of the parameter should be added to the top. If the stack is not full, the pointer attribute should be incremented and the parameter should be added to the element indicated by the pointer. The Push subroutine returns no value.
* Pop requires no parameters but returns an integer. If the stack is empty, a value of -1 should be returned. Otherwise the value in the element indicated by pointer should be returned and pointer should be decremented.

In PlayGame, create an instance of the History class called past. When the program deals with a ‘go’ command, this instance should be passed to the go subroutine as an additional parameter.

In go, modify the parameters to accept the instance of the History class. As a player is moving, Push their current location onto the stack before they move to a new one. If the instruction ‘go back’ is entered, send the player to their previous location using a call to the Pop subroutine of History. If the call to Pop returns a value of -1, display the text ‘You cannot go back’.

|  |
| --- |
| **Evidence you need to provide:**   * Your SOURCE CODE for the new History class * Your amended SOURCE CODE for PlayGame * Your amended SOURCE CODE for Go * One screenshot showing the output that results from the following:   + Type ‘flag2’ when prompted for a file name   + At the next prompt, type ‘go east’   + At the next prompt, type ‘go back’   + At the next prompt, type ‘go back’ a second time |

## **Task 12**

This question relates to the Character class, the PlayGame subroutine, the LoadGame subroutine and a new subroutine EatItem.

Modify the Character class so that each character in the game has a Health property.

Modify the LoadGame subroutine so that each character’s initial Health value is 10.

Create a new subroutine called EatItem which requires as parameters the characters List, the items List and a string called itemToEat. If the itemToEat string matches the name of an item in the player’s inventory, and that item’s Status attribute contains the text ‘edible’, increase the player’s Health property by 5 and remove the eaten item from the List. If any of these conditions are not met, the text ‘You cannot eat that’ should be displayed. If an item is successfully eaten, the game should display the current player’s Health.

Modify PlayGame so that the player’s Health is displayed immediately after the call to DisplayGettableItems. Include a call in PlayGame to EatItem when the player enters an instruction beginning with ‘eat’.

|  |
| --- |
| **Evidence you need to provide:**   * Your amended SOURCE CODE for the Character class * Your amended SOURCE CODE for LoadGame * Your SOURCE CODE for the new EatItem subroutine * Your amended SOURCE CODE for PlayGame * One screenshot showing the output that results from the following:   + Type ‘flag1’ when prompted for a file name   + At the next prompt, type ‘eat apple’   + At the next prompt, type ‘eat apple’ a second time |

## **Task 13**

This question relates to TakeItemFromOtherCharacter.

Currently, if you win a dice game against an opponent and mis-type the item you want to take from them, you are told that you do not receive an item and you are given no opportunity to re-type. Modify TakeItemFromOtherCharacter so that if you request an item that they do not have, the message ‘They don’t have a \_\_\_\_\_ - try again’ is displayed (with \_\_\_\_\_ replaced by the item the player tried to take). The player is then asked again which item they want to take, with the available items re-listed. This will continue to happen until the player has entered an item that the other player possesses, at which point the item will move to the player’s inventory as normal.

|  |
| --- |
| **Evidence you need to provide:**   * Your amended SOURCE CODE for the TakeItemFromOtherPlayer class * One screenshot showing the output that results from the following:   1. Type ‘flag2’ when prompted for a file name   2. At the next prompt, type ‘playdice guard’   3. Type ‘6’ and ‘6’ for the minimum and maximum at the second and third prompts   4. Repeat the previous two steps if the result is a draw   5. When asked to choose an item, type ‘box’   6. At the next prompt, type ‘jar’ |

## **Task 14**

This question relates to CheckIfDiceGamePossible.

Currently, if a dice game is not possible, the program outputs the message ‘You can’t play a dice game’. This occurs whether the player has no die, the opponent has no die or the player and the opponent are not in the same room. Modify CheckIfDiceGamePossible so that the following messages are output:

1. If the player has no die, the output should read ‘Player has no die’
2. If the player and the opponent are not in the same room, the output should be ‘\_\_\_\_\_ is not here’ (with \_\_\_\_\_ being replaced by the name of the other character)
3. If the opponent has no die, the output should be ‘\_\_\_\_\_ has no die’ (with \_\_\_\_\_ being replaced by the name of the other character)

In all cases, these messages should still be followed by the message ‘You can’t play a dice game’. In the event that more than one of these conditions are met, only one should be displayed, with conditions checked in the order in which they are numbered above. So if the player has no die, the program will not check that the opponent is present or that the opponent has a die. If the opponent is not present, no check will be made that they have a die.

No code should be modified besides the content of CheckIfDiceGamePossible.

|  |
| --- |
| **Evidence you need to provide:**   * Your amended SOURCE CODE for CheckIfDiceGamePossible * One screenshot showing the output that results from the following:   + Type ‘flag1’ when prompted for a file name   + At the next prompt, type ‘playdice guard’   + At the next prompt, type ‘get red die’   + At the next prompt, type ‘playdice guard’ |

## **Task 15**

This question relates to DisplayInventory.

Currently, when the user enters ‘examine inventory’, a list of the names of the items being carried by the player is displayed. Modify DisplayInventory so that descriptions are displayed alongside the names. Each description should be displayed in round brackets. Additionally, the opening brackets of the descriptions should line up in a column, regardless of variations in an item’s name.

|  |
| --- |
| **Evidence you need to provide:**   * Your amended SOURCE CODE for DisplayInventory * One screenshot showing the output that results from the following:   + Type ‘flag1’ when prompted for a file name   + At the next prompt, type ‘examine inventory’ |