Lab 7 [Inventory]

In this workshop, we will continue our development of a text based game "Code Quest!". This workshop focuses on item inventory using parallel arrays.

LEARNING OUTCOMES

Upon successful completion of this workshop, you will be able to

- Code a program that uses parallel arrays
- Write code which searches and updates arrays

SPECIFICATIONS

Part 1:

Write a program which acts as an inventory system for CodeQuest! This program stores a list of item IDs and item quantities. It features a series of utility functions for managing. This program requires the following four functions:

`int Find_Item(int id[],int size,int item);` This function performs a linear (naïve) search over the inventory array. This function receives an integer array of item `id's`, an integer for the `size` of the array, and an integer for the desired `item`. This function searches through the array for the desired item and returns the index of the item if it is found, -1 if it is not found.

`int Add_Item(int id[],int quantity[],int size, int item);` This function adds an item to the inventory of the player. It receives an int array of item `id`**'s**, an int array of item `quantity's`, an int for the `size` of the arrays, and an int for the `item` being added. This function searches through the `id` array to find if the `item` exists in the array (use `Find_Item` function above). If it finds the `item`, it increases the associated element in the `quantity` array by one. If it does not find the item, it searches the `id` array for an item in the array with an associated `quantity` of zero. If it finds one, it changes the id to the id of the item being added and sets the associated quantity to 1. If the item was able to be added (by replacing a zero quantity item or increasing an already existing one) this function returns 1 (item successfully added), otherwise it returns 0 (inventory already full).

`int Use_Item(int id[],int quantity[],int size,int item);` This function removes an item from the inventory. This function receives an int array for the item `id`, an int array for the item `quantity`, an int for the `size` of the array and an int for the `item` id of the item being used. This function searches the inventory for the desired item, if the item's

quantity is greater than 0, it reduces the quantity by one and returns 1 (success), otherwise, it returns 0 (unable to find item with non-zero quantity).

`void Print_Item(int item):` This function prints the item name associated with an item ID. It receives an int for an **`item`** id. This function simply prints the item name associated with the item id parameter. The associations are as follows:

0 - Potion, 1 - HP Booster, 2 - Strength Booster, 3 - Defense Booster, 4 - Intelligence Booster

This function assumes the item ID is valid.

Part 2:

Once the above four functions are implemented, write a program that simulates a basic inventory system. The program creates two integer arrays for item **`id's`** and item **`quantities`**. The arrays should both be 3 elements in size and all values to be initialized to zero. Present the user with a menu with the following options:

1. Get New Item
2. Show Inventory
3. Use Item
4. Quit.

The program loops until 4 is selected.

If 1 is selected, the program generates a random integer between 0 - 4 (including 0 and 4) and adds it to the inventory. Use **`Add_Item`** function above.

If 2 is selected, display the item ID, item name and quantity to the user. Please note that the code for this option is given below.

If 3 is selected, the user is prompted for an item id. The program then reduces the quantity of that item in the inventory if it exists. Use **`Use_Item`** function above.

```
//code for option 2:

int i;
for(i=0;i<3;i++)
{
     printf("%d. ",id[i]);
     Print_Item(id[i]); //Use the function explained above.
     printf("%d\n",quantity[i]);

}
```

Sample output:

The output of a typical run-through of your program should look like this (user input highlighted in green). Items are randomly generated thus outputs will not match exactly:

```
1. Get New Item
2. Show Inventory
3. Use Item
4. Quit

Select Option: 1

New Item Added!

1. Get New Item
2. Show Inventory
3. Use Item
4. Quit

Select Option: 1

New Item Added!

1. Get New Item
2. Show Inventory
3. Use Item
4. Quit

Select Option: 1

New Item Added!

1. Get New Item
2. Show Inventory
3. Use Item
4. Quit

Select Option: 1

New Item Added!

1. Get New Item
2. Show Inventory
3. Use Item
4. Quit

Select Option: 1

Inventory is already full!

1. Get New Item
2. Show Inventory
```

```
3. Use Item
4. Quit

Select Option: 2

0. Potion X2
4. Intelligence Booster X1
2. Strength Booster X1

1. Get New Item
2. Show Inventory
3. Use Item
4. Quit

Select Option: 3
Select item: 1
You don't have that item!

1. Get New Item
2. Show Inventory
3. Use Item
4. Quit

Select Option: 3
Select item: 0
Item used!

1. Get New Item
2. Show Inventory
3. Use Item
4. Quit

Select Option: 3
Select item: 4
Item used!

1. Get New Item
2. Show Inventory
3. Use Item
4. Quit

Select Option: 2

0. Potion X1
2. Strength Booster X1

1. Get New Item
2. Show Inventory
3. Use Item
4. Quit

Select Option: 4
```

Part 3: (BONUS)

BUILDING THE GAME (OPTIONAL)

If you have completed all previous game labs, you may complete this section in order to merge your current lab with you previous labs.

Copy the functions (and prototypes) into your codeQuest source code file. Move the inventory array init code to where the character creation code is. Add a new option to the overworld menu for "Open Inventory". When this option is selected, display a similar menu as the one in this lab (remove the "Get item" option). When the "use item" option is chosen, the function should modify the user's stats based on what item is chosen (potion heals the player's health, others increase associated base stats by one.

The battle sequence should be slightly modified so that if the user wins a battle, a random item is added to the user's inventory.

Prepare a Typescript

Create a typescript on the remote Linux host using the following commands:

```
+ At the prompt, type: script w7.txt
+ At the prompt, type: whoami
+ At the prompt, type: cat w7.c
+ At the prompt, type: gcc w7.c –o w7.out
+ At the prompt, type: w7.out
+ At the prompt type: exit
```

This will produce a typescript named **"w7.txt"** in your current directory. Transfer a copy of this file to your local computer.

SUBMISSION

Upload your typescript file to BlackBoard:

- Login to BlackBoard
- Select your course code
- Select Workshop 7 under Workshops
- Upload **`w7.txt`**
- Write a short note to your instructor
    - Under “Add comments”, add a sentence or two regarding what you think you learned in this workshop in the notes textbox
    - press "Save Changes"
- When ready to submit, press "Submit". Note you can save a draft until you are ready to submit.