Exercise 1: BAG/Shop

1- Create a class that represents **bag** objects in a shop. For each bag you store:
   a. **Size** (**float**)
   b. **Slots (number of slots in the bag)**
   c. **Price**

2- Member Functions of class **Bag**:
   a. **Constructor**: takes the following 2 parameters with these default values:
      i. size: **15.6**
      ii. number of Slots = **5**

      The constructor should initialize price, to **0**.

   b. **Setters for all members**: If the passed parameter is valid, the data member should be changed, and the function should return **true**. Otherwise, the member variable **won't be affected,** and the function should return **false**.)

   c. **Getters for all members**

   d. **Compare:** this function should compare the current bag with another bag passed to the function. It returns **1** if the current bag is *bigger* than the passed bag, and returns **-1** if the passed bag is bigger than the current one. If the sizes are equal, the bag with a greater number of slots is considered bigger. Otherwise, the function returns 0 as when both size and slots are equal.

   e. **PrintInfo**: Print all the details of the bag in the following format.

   f. **ReadInfo**: Reads the information of the bag from the user

3- Create a class **Shop** to represent bags shop with members:
   a. **BagList** (Array of 50 **pointers to Bag**)
   b. **Quantity** (Number of bags available in the shop).
   c. **Total profit** (Total profit for the sold bags)

   d. **Constructor**: to initialize Shop members.

   e. **IsAvailabe**: This function takes bag size and no. of slots returns the index of the first bag with the passed parameters. Otherwise, it returns -1.

   f. **AddBag**: Add a new bag to the BagList if there is a vacant place. Then reads the data of the new bag from the user.

   g. **GetBag**: takes and index and returns a pointer to the bag stored at this index if any. Otherwise, returns NULL. If index is out of range, function returns NULL too.

   h. **Sell:** this function takes bag size and no. of slots and searches for the first bag with the passed parameters. If found, it removes (sells) the bag from the list and adds its price to the total profit.
   [Note]: Call **IsAvailable** function to check the availability of the required bag.

   i. **PrintAllBags**: Prints information of all bags in the shop

4- Write a program (**main function**) that:
   a. Create a Shop object **S**
   b. Add 3 bags to the shop with arbitrary values for their members.
   c. Print the info of all bags.
   d. Declare Bag **Pointer Bp**.
   e. Dynamically allocate a bag using the following constructor parameters:
   (Size = **15.6**, Number of slots = **7**) and make **Bp** point to it.
   f. Set the price of **Bp** to **130**
   g. Call GetBag and point to the returned Bag by pointer **ptr**.
   h. Compare the object pointed to by **Bp** with the object pointed to by **ptr** and print the comparison result"
   i. Sell the 2nd bag in the shop.
   j. Print the info of all bags.