Assignment Instructions – please read carefully.

1. This is a group assignment. You should work in groups of **at most two students**.
   - You should work on the whole assignment together.
   - You should **not** just partition the work between you.
   - You should upload only one submission to canvas.
   - You may check your team member on Canvas under People → Assignment_Groups.
2. Written Parts:
   - Please type your answers and hand them in as a **PDF** file through Canvas.
   - Please type your answers for parts 1 & 4 and hand them in as a **single PDF** file through Canvas.
   - You should use a diagramming platform or program to create your class diagram.
   - Please include your full names on the first page of the PDF.
   - **DO NO include the PDF file in the ZIP file**, please submit it separately on Canvas.
3. Coding Parts:
   - Hand in all program source files softcopy using Canvas
   - Be sure that they properly execute. Failure to do so will mean that your program is not graded.
     Note: assignments are graded using the terminal and `C++11` standard.
   - All source files should be compressed into a *ZIP* archive.
   - Use the following naming format:
     `<firstNameLastname1>_<firstNameLastname2>_asmt<#>.zip`.
     Example: `SaraElAlaoui_JaneDoe_asmt2.zip`
   - You must use the code provided on Canvas.
   - Your ZIP file should include: (1) the folder `LinkedBagDS`, (2) `AmazonMain.cpp`, (3) all the classes `.cpp` and `.h` files, and (4) sample inputs/outputs `input01.txt` and `output01.txt`, if you choose to do the extra credit part.
4. **Please refer to the university integrity policy on the syllabus, and remember that you should be able to explain and reproduce any code you write and submit as part of this assignment.**

Assignment Goals and Objectives

1. Programming methodology:
   - Analyze problem requirements.
   - Generate a class diagram from the problem requirements.
2. C++ implementation:
   - Implement classes, including base and derived classes.
   - Analyze the linkedBag implementation of ADT Bag (similar to List).
   - Implement functions to manipulate LinkedBags.

Problem definition

You will implement a simplified version of Amazon. Your program should include:

1. a complete implementation of the main function,
2. definition and implementation files (`.h` and `.cpp`) for each class needed in the program,
3. at least one instance of inheritance,
4. an implementation of two additional functions to the linked-list implementation of the ADT Bag (LinkedBag), and
5. at least one instance of using each of these functions in your program.

Application requirements – Amazon340

This is a very simple Amazon-like application that only implements one main aspect: managing a vendor's products.

Your application should allow a user to perform the following tasks:

1. create a profile: username, email address, password, short bio, and profile picture (typically, a string storing the path to an image saved somewhere),
2. display their information (it should not display their password),
3. modify their password, (display the new password for testing purposes)
4. create a new product,
5. display all their products,
6. display their k^{th} product (e.g. their 3rd product),
7. modify a product: the vendor needs to specify the index of the product they want to modify. They should then be able to modify the name and description.
8. Sell a product (using the specification below): the user needs to specify the index of the product they want to sell, and
9. delete a product.

The main function that displays this menu has been provided in the starter code. Your task is to call the right function from your implementation to perform the given choice.

Products: Each product has a name, description, rating, and a count of how many times it has been sold. The application has two types of products: Media and Goods.

1. Media
   - In addition to the information above, media also has a `type` and `target audience`.
   - Selling media: When you sell a media, there is no quantity to update, but you should provide the user with a one time access code.
2. Goods:
   - In addition to the general information of a product, goods will have an `expiration date` and a `quantity`.
   - Selling Goods: Unlike media, when selling goods, you need to check if you have enough items in stock. If you are able to sell the good, you should update the quantity accordingly; otherwise, you should inform the user that the product is sold-out.

Vendor products: A vendor's profile can only have one list containing all their products (i.e. do not create two lists, one for Goods and one for Media).

Additional requirements: You client/interviewer **requires that you use LinkedBag** instead of the C++ standard libraries such as `vector`. They ask you to add two new functions (defined in **Part 3** of the assignment) to `LinkedBag.cpp` to match the application's requirements. As such, any list of objects in your program should be stored and manipulated using the data structure LinkedBag.

Part 1 – 25 points, Due February 27th
Generate a Class Diagram for your application using Unified Modeling Language (UML) notations.

- It should include the following classes:
  1. Amazon340,
  2. Vendor,
  3. Product,
  4. Goods and
  5. Media
- Read the program requirements carefully to extract members of each class.
- The class diagram should be created using a diagramming software. Points will be deducted if the class diagram is hand-drawn.

Note 1: Linked Bag is a data structure that is separate from your program, so it is **not to be included in the class diagram**.
Note 2: `https://draw.io/` is a great platform for creating diagrams.

Part 2 – 50 points Implement the classes identified in the class diagram. Your program should include the following:

1. Main program source file: The main program is provided, and it includes a skeleton of the `main` function. The function prints out a menu with various actions the user can take.
2. Header and source files for each class. Implementations should match **the class diagram provided on February 27th**.

Part 3 – 20 points
The interface for Bag ADT is provided to you in The folder `LinkedBagDS`. Its linked-list implementation is also included in the same folder. You should only modify `LinkedBag.cpp` in this part.
Add the implementation of the following functions:

1. `bool appendK(const ItemType& newEntry, const int& k)`: adds the element to the Kth position in the LinkedBag. If K is out of range, add to the end.
2. `Node<ItemType>* reverseFindKthItem(const int& k)`: returns a pointer to the k^{th} element from the and of the LinkedBag.

Part 4 – 15 points Choose 2 principles of Object-Oriented Programming and describe (1) how you incorporate it in your program and (2) why it is important.

Extra Credit – 5 points

You should provide one non-trivial test case to show that you have tested your program for correctness. your test case should include sample inputs and expected outputs.
Sample input should be a sequence of menu options a user would choose and the sample output would be the expected behavior of the program or next step.

Save you sample inputs in `input01.txt` and its expected output in `output01.txt`. Include them in your ZIP file.