CO1107 Algorithm, Data Structure & Advanced Programming - Workshop Week 3

Task 1:

Modify your program from Task 3 of week 2, so that it repeatedly asks the user to select from the following options:

1) Print - prints the list in the order it is currently sorted (or unsorted order if it has not yet been sorted).
2) Sort on Distance - sorts the current list in ascending order of distance.
3) Sort on Price - sorts the current list in ascending order of price.
4) Quit - program stops.

Sample output

```
Enter a file name: Tiny.txt
Enter choice (1): Print / (2): Sort on Distance / (3): Sort on price / (4):Quit : 1

120 Miles, £ 150.12
140 Miles, £ 180.1
70 Miles, £ 250.02
99 Miles, £ 398.72
144 Miles, £ 205.42

Enter choice (1): Print / (2): Sort on Distance / (3): Sort on price / (4):Quit : 2

70 Miles, £ 250.02
99 Miles, £ 398.72
120 Miles, £ 150.12
140 Miles, £ 180.1
144 Miles, £ 205.42
Enter choice (1): Print / (2): Sort on Distance / (3): Sort on price / (4):Quit : 3

120 Miles, £ 150.12
140 Miles, £ 180.1
144 Miles, £ 205.42
70 Miles, £ 250.02
99 Miles, £ 398.72

Enter choice (1): Print / (2): Sort on Distance / (3): Sort on price / (4):Quit : 4

Quitting . . .
```

Task 2: Write a python function subsetOf that accepts two lists of integers L and M as arguments, where M is a subset of L (assume L and M do not contain any duplicate values) , and then returns a list of zeroes and ones, K, such that K[i] = 1 if and only if L[i] is found within the list M.

For example: if L = L=[2,17,12,5,66,20,7] and M=[2,12,66] then your function should return the list [1, 0, 1, 0, 1, 0, 0] which represents that M contains the items found in L at positions 0; 2 and 4.

Task 3:

Write a Python function, **duplicate,** which takes two lists sorted in ascending order as input and returns a list of items that appear in both lists.

Task 4:

Using only stack operations, find the largest item in a stack.

Task 5: (will not be assessed as part of lab effort)

Given the file below containing information about some product. Each line in the file consists of a product ID followed by a tab followed by a product name.

For example, product.txt contains:

```
3521	books
6421	shoes
3212	computers
6631	printers
5438	shirts
```

1) Write a program to create a list of productID / productName pairs based on the file contents.
2) Write a function binarySearch that takes 2 inputs: a list L(created in part 1) and a user input for the name of the product, to find the productID of the given productName. If the productName is not in the database, your program should print "Not Found", otherwise it should print the associated productID.

Sample output:

```
Enter your filename: product.txt
Enter the Product Name: computers
Product ID is : 3212

Enter the Product Name: scanner
Not found
```