THE FINAL HOMEWORK

You have “accidentally” consumed a forbidden shroom. Below are the voices and thoughts you experience during your “trip”:

You are now an expert in Hyper Intelligence (HI).

You see a lot of fruits around you: apple, banana, cat …

Wow, very cool. How about sorting them!

“(You) No, they are not ordinal! There is no natural order in them!”

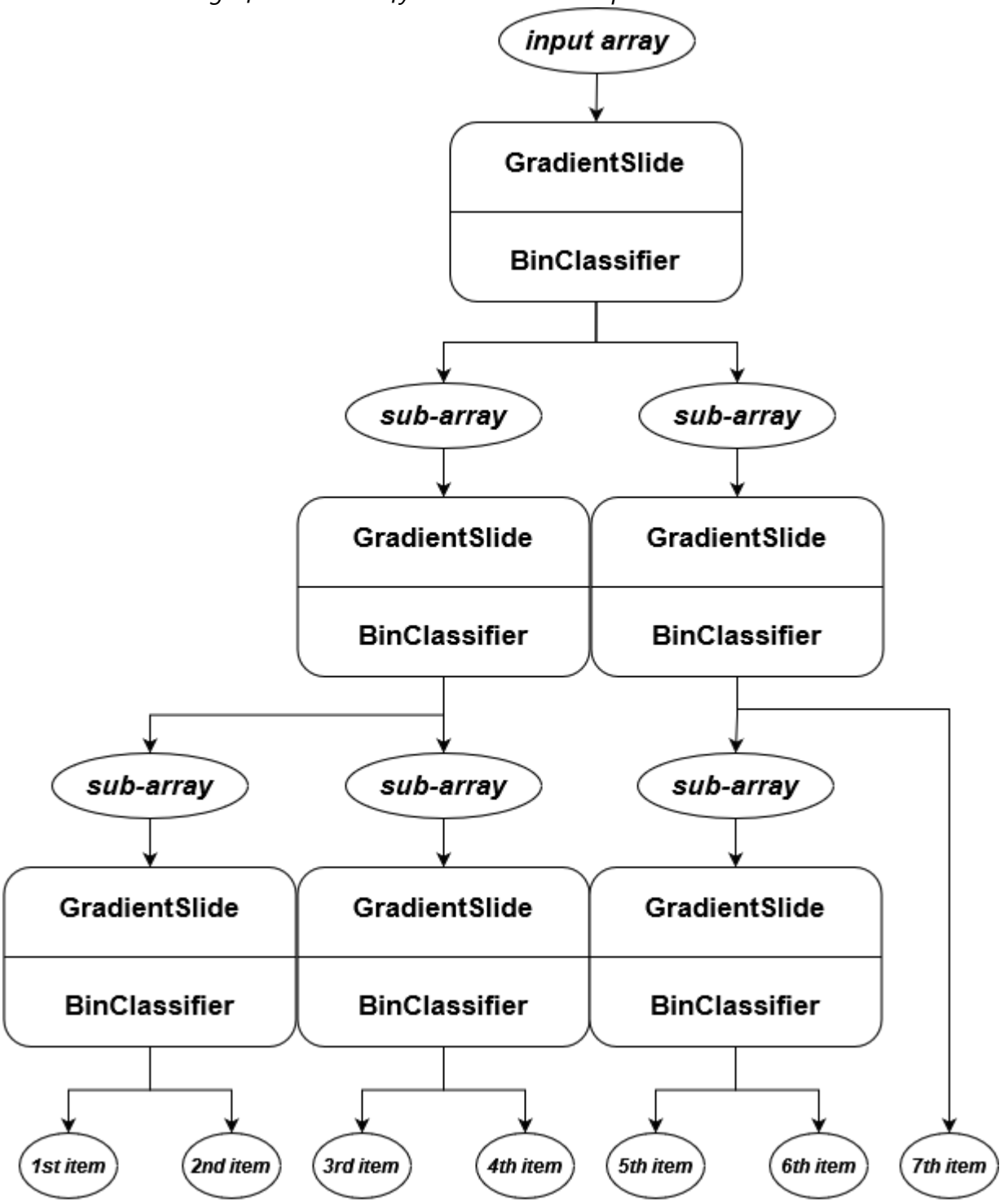
Listen, you are a HI expert. You are HI! Use your power!

“(You) Ok, ok! But how? Please tell me, *mysterious people that sounds a lot like Morgan Freeman*”

You see, fruit grows on tree. Use tree to help you! And please don’t call me *Morgan Freeman*.

You think in silence for a while, and now an image depicting a *Hyper Learning Architecure* is in your head

*** a tree consisting of a BinClassify+GradientSlide operation ***



- BinClassify splits item into two groups
- GradientSlide applies a simple transformation

"(You) Ah! It's just like some uhh LightningSort algorithm! I can just process them recursively!"

The *not Morgan Freeman* smiles and fades into the background as you slowly wake up from the "trip".

It was a surreal experience. You try to recall the technical details to see if Hyper Intelligence can be the next cool kid in the tech world. You tried your best and recorded down some “fruits” and their “feature hypervectors”, you also managed to recall their “pre-trained hyperweights”. These are the information you piece together:

```cpp
/*
 * Hyper Learning Framework 1.0
 *  note to self: this is not DEEP learning
 *  note to self2: shroom is not bought on Guangfu Rd. Section 1
 *  it looks a lot like something I have learnt in CS 235100...?
 */
struct Fruit {
    std::string      name;
    std::vector<int> hyperFeatures;
};

struct GradBinHyperWeight {
    int                featureIdx;
    double             gradSlider;
    int                binThreshold;
    GradBinHyperWeight *backwardContinuation = nullptr;
    GradBinHyperWeight *forwardContinuation = nullptr;

    bool shouldBackward(Fruit& f) const
    {
        return f.hyperFeatures[featureIdx] < binThreshold;
    }
    void applyGradSlider(Fruit& f) const
    {
        f.hyperFeatures[featureIdx] = (int) ((double) f.hyperFeatures[featureIdx] * gradSlider);
    }
}
```

To see if it actually works, you decided to write a program that will perform the sorting on the testcases. It should print out the fruits in the sorted order, and also the last used *HyperFeature* just to check if it makes sense.

Note: You may use `<string>`, `<vector>` and `<functional>` for this homework.

Input

For each testcase:

```
n (Number of Fruit)
m (Number of Hyperfeature)
[n x <FruitDescriptor>]
o (Number of GradBinHyperWeight)
[o x <GradBinHyperWeightDescriptor>](array representation of tree)
```

FruitDescriptor

```
name
[m x <HyperFeature value, int>]
```

> GradBinHyperWeightDescriptor

```
featureIdx
gradScaler
binThreshold
```

> array representation of tree:
>
> > 1st item is the root
> > 2i-th item is the left child of i
> > 2i+1-th item is the right child of i

Output

```
<fruit name> <value of last used hyperfeature>
```

Sample

Sample Input:

```
4
1
Dragonfruit
10
Apple
3
Cat
78
Mango
99
3
0
1
40
0
1
5
0
1
80
```

Sample Output:

```
Apple 3
Dragonfruit 10
Cat 78
Mango 99
```