Create – Applications From Ideas
Written Response Submission Template

Please see Assessment Overview and Performance Task Directions for Student for the task directions and recommended word counts.

Program Purpose and Development

2a)

The program shown in the video is written in Java. The purpose of the program is to give the information of the longest fence built by wood pieces with the same length of one but different height. Exactly two wood pieces are vertically nailed together to make a board; the boards with the same height are horizontally nailed to make the fence. The program requires two lines of input: the first line gives the total number of available wood pieces; the second line gives the height of each individual wood. The program outputs the length of the longest fence and the number of different heights the longest could have. The user in the video ran the program twice to get the information about the fences, and chose to look at the additional information about possible height.

2b)

The NailedBoard Program was developed in stages:

1) Read the input data of wood pieces and sort it.

2) Identify different values and their corresponding frequencies.

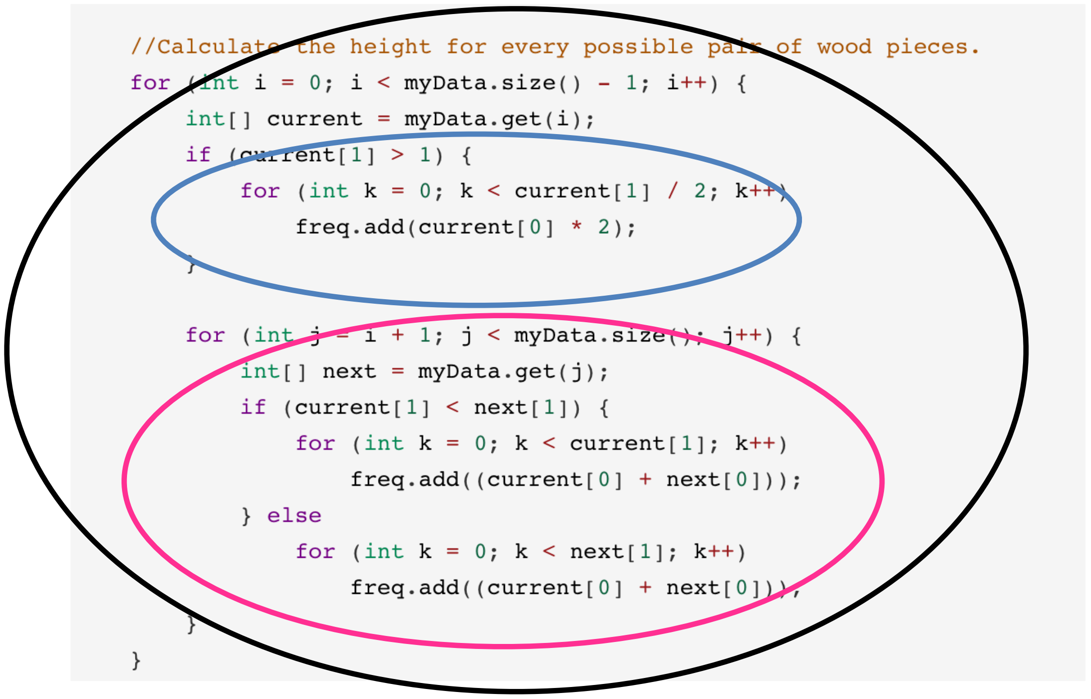
3) Calculate the height for every possible board made by two wood pieces.

4) Determine the most frequent height values and output the information.

I encountered a problem during the stage three. The program at first overcounted the pairing due to the various frequency of each value. For example, if the input is “1 3 4 4”, the “1” and “3” can be paired to only one of the “4”s. To solve this problem, I restricted the loop iteration under the maximum times of pairing.

The first version of the program only gave the information of the length and number of possible heights for the longest fence. I realized an opportunity to enhance users’ experience of the program, which was enabling people to get the exact height for each possible fence. As a result, I added a function to print out each value so that people can see the pairing directly.

2c)



```
//Calculate the height for every possible pair of wood pieces.
for (int i = 0; i < myData.size() - 1; i++) {
    int[] current = myData.get(i);
    if (current[1] > 1) {
        for (int k = 0; k < current[1] / 2; k++)
            freq.add(current[0] * 2);
    }

    for (int j = i + 1; j < myData.size(); j++) {
        int[] next = myData.get(j);
        if (current[1] < next[1]) {
            for (int k = 0; k < current[1]; k++)
                freq.add((current[0] + next[0]));
        } else
            for (int k = 0; k < next[1]; k++)
                freq.add((current[0] + next[0]));
    }
}
```

The big algorithm circled in black oval is the fundamental method to calculate the height for every possible pair of wood pieces. Since the frequency for each occurred value is different, a wood piece may face two conditions of pairing: with the same value or a different value. The blue algorithm is responsible for pairing the number if it occurs more than one time. For example, for “4,4,4”, exactly one board can be made by any two “4”s among the three. The red algorithm is constructed to pair a wood piece with a different height, taking into account the maximum frequency threshold. For example, if there are three “2”s and one “3”, only one of the “2”s can be paired to “3”. The two small algorithms are combined together to generate the possible heights for further use.

2d)

```java
public static int count(int a, int b, int i1, int i2){
    boolean same = true;
    int occur = 0;
    while (same) {
                if (a < b - 1 && data[i1] == data[i2]) {
                    occur = occur + 1;
                    i++;
                } else
                    same = false;
            }
    return occur;
}
```

The counting process was needed several times during the frequency checking. To reduce the complexity of the program, I created a method called “count” to count the frequency of each value. This method was used in the input frequency checking as well as the pairing sum frequency checking. Since the frequency of each value was unknown, the method used a while loop to continuously do the counting.