Trick or Treat!

Halloween is coming soon and you've been asked to create a program to help parents and children get just what they want. Parents want to give children an upper bound on the amount of candy they can receive and children want as much candy as they get without upsetting their parents.

Fortunately, in our neighborhoods, we know in advance exactly how many pieces of candy each home hands out to the children. A child has to take all the candy given by the home so they don't seem rude, and they can't throw away or eat any candy on the way. Children also have to stop at every home in the sequence of homes.

Question:

Given these conditions write a program to find the best sequence of homes for children to visit with following input/output.

Input:

Input will be read from an ASCII text file (input.txt). The input will consist of information about one neighborhood. The first line contains a single integer, *homes*, $0 < homes \leq 10{,}000$, that represents the maximum number of homes on the block.

The next line contains a single integer, *max*, $0 \leq max \leq 1000$, representing the maximum number of pieces of candy that the child may collect. There will then be *homes* lines, each containing an integer, *pieces*, $0 \leq pieces \leq 1000$, representing the number of pieces given at each home. Homes are numbered consecutively starting at 1.

Output:

If there is no way to select one or more consecutive homes where the sum of pieces of candy is $\leq max$ on a single line, print "`Don't go here`". Otherwise, in the format shown below, for the sequence of homes that yield the largest number of pieces of candy $\leq max$, include the number of the first home to visit, the number of the last home to visit, and the sum of pieces of candy.
If there is more than one such sequence of homes, select the one with the lowest numbered first home.

Sample Input:

```
5
10
2
4
3
2
1
```

Output Corresponding to Sample Input:

```
Start at home 2 and go to home 5 getting 10 pieces of candy
```

Subsidiary question if you are familiar with OpenMP:

Parallelize the program with OpenMP for multi-core execution.