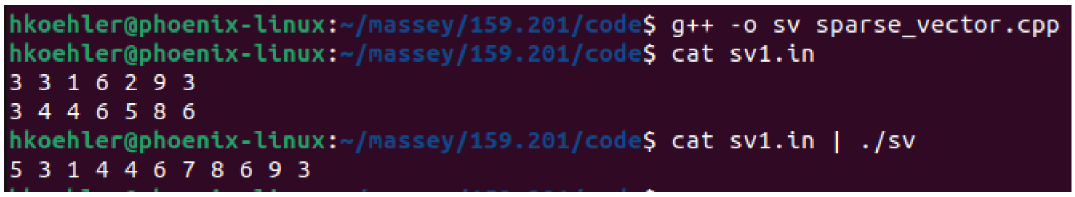
159.201 Algorithms & Data Structures
Assignment 1

A vector is considered *sparse* when most of its entries are zero. Sparse vectors can be represented more efficiently by storing only non-zero values and their positions.

In this assignment you need to implement vector addition for sparse vectors. You need to write and submit a program (in C/C++) that reads two sparse vectors from standard input, adds them, and prints the resulting sparse vector to standard output. The format of sparse vectors to be used for I/O is the following:

$$n \text{ index}_1 \text{ value}_1 \text{ index}_2 \text{ value}_2 \ldots \text{ index}_n \text{ value}_n$$

Here n denotes the number of non-zero values, value$_i$ the i^{th} non-zero value and index$_i$ its position in the vector. Non-zero values must be listed in ascending order of position, for both input and output. Values are positive or negative integers, small enough to be stored and added as `int`. A usage example is shown below.



```
hkoehler@phoenix-linux:~/massey/159.201/code$ g++ -o sv sparse_vector.cpp
hkoehler@phoenix-linux:~/massey/159.201/code$ cat sv1.in
3 3 1 6 2 9 3
3 4 4 6 5 8 6
hkoehler@phoenix-linux:~/massey/159.201/code$ cat sv1.in | ./sv
5 3 1 4 4 6 7 8 6 9 3
```

Assuming zero-based indexing, the two vectors added here are

```
  0 0 0 1 0 0 2 0 0 3 0 0 0 ...
+ 0 0 0 0 4 0 5 0 6 0 0 0 0 ...
= 0 0 0 1 4 0 7 0 6 3 0 0 0 ...
```

Your code must represent sparse vectors in "sparse format" throughout, meaning memory consumption must only depend on the number of non-zero values (and not on the maximum index). Other than that, you may use any data structure you like.

You can use CodeRunner to test your work, but must submit your assignment as usual in Stream.