Lab 9

Part 0. For the parts 1 and 2 below, place the function headers in bitvector.h, the function code in bitvector.c, and the main function to test your code in main.c. Write a makefile separately compiling bitvector.c and main.c; and then linking them to make the executable.

Part 1. Bit vectors are used to provide dense storage and allow fast manipulation of the entire group of bits at once. In a bit vector, each individual bit is designated to represent a particular value and a bit is turned on to indicate its corresponding value is contained in the set. In this problem, we have sets drawn from the range [1-9]. A set is represented using an unsigned short (16 bits). The bit at the nth position (counting from least significant bit at the 0th position) is designated to represent the value n. The bits in the positions 1 to 9 represent the set membership and the other seven bits of the short are ignored. Below is a few example sets and their corresponding bit vectors.

0000000000000000 // {}

0000001010100100 // {2 5 7 9}

0000000000001110 // {1 2 3}

Write the makeSet function to create a bit vector set from an array. The arguments to the function are an array of integer values and the array count. The function returns an unsigned short that represents a bit vector set of the values from the array. The bits in positions 1-9 of the returned result mark the set membership, the remaining bits are zeros. You may assume that no value in the array will be outside the range 1-9.

unsigned short makeSet(int values[], int nvalues)

Part 2. Now write the isSingle function to manipulate these bit vector sets in efficiently comput- ing a result needed when solving a Sudoku puzzle. The goal of Sudoku is to assign every cell in the grid a digit from 1 to 9 subject to the constraint that each digit can be used only once in each row, column, and block. (You can read about Sudoku in wikipedia if you're curious, but for the purposes of this problem, you don't need to know any more details). The isSingle function is given three bit vector sets representing the digits already used in the row, column, and block for a cell. The possibilities for a cell consist of only those digits that are unused; any digit that is already used in

the row, column, or block is not a possibility. The function isSingle returns true if the already used digits force only one option for this cell (number of possibilities is exactly 1) and false otherwise (i.e. number of possibilities is 0 or 2 or more).

bool isSingle(unsigned short used in row, unsigned short used in col, unsigned short used in block)