Files to submit: knapsack.s
Time it took Matthew to complete: 1.5 hours (This one is a pain in the butt to debug so start on it early)

- All programs must compile without warnings when using the -Wall and -Werror options
- Submit only the files requested
  - Do **NOT** submit folders or compressed files such as .zip, .rar, .tar, .targz, etc
- If submitting in a group on Grade Scope please make sure to mark your partner.
  - Only one of you has to submit there
- Your program must match the output exactly to receive credit.
  - Make sure that all prompts and output match mine exactly.
  - Easiest way to do this is to copy and paste them
- All input will be valid unless stated otherwise
- Print all real numbers to two decimal places unless otherwise stated
- The examples provided in the prompts do not represent all possible input you can receive.
- All inputs in the examples in the prompt are underlined
  - You don't have to make anything underlined it is just there to help you differentiate between what you are supposed to print and what is being given to your program
- If you have questions please post them on Piazza

1. Write a program called **knapsack.s** that solves the 0-1 knapsack problem **recursively**. In the knapsack problem you have a knapsack that can hold W weight. You also have a collection of items that each have their own weight w_i and value v_i. The goal is find the set of items that maximizes the amount of value in the knapsack but whose weight does not exceed W.
    1. This program should be callable from C and have the following signature
        1. `unsigned int knapsack(int* weights, unsigned int* values, unsigned int num_items, int capacity, unsigned int cur_value)`
        2. This function should calculate and return the maximum value knapsack
        3. You may not have a data section
        4. This function must be implemented **recursively**
        5. Pay very careful attention to the **types** in this function as it will affect which machine instructions you should use. Hint: it will affect the jump instructions you use
        6. You have been provided with a C file called knapsack.c that implements this function and should give you a good starting point
            1. If you want an extra challenge try solving the problem without looking at knapsack.c as this problem boils down to just finding the optimal *combination* of items
        7. You will find the `leal` instruction very helpful for this problem
    2. You have also been given a file called main.c that will take as a command line argument the name of a file containing a knapsack problem.
        1. Please see the comments in main.c to see how these files are structured
        2. Your function must be callable from this file
    3. You have also been given a makefile that should compile your program. Your program **MUST** be able to be compiled by this makefile.
        1. For those of you running 64 bit versions of Linux you may need to install the 32 bit binaries.
        2. The command to install on Ubuntu is: `apt-get -y install gcc-multilib`

    4. Example:
        1.
```
cat Tests/0-test.txt
100
4
43 43
3 38
5 17
18 25

./knapsack.out Tests/0-test.txt
123
```