Assignment 2 Part 2

Like any program there are a LOT of details that need to be addressed. You should have discussed some of them for the `product-digit-sum` program in a recent class meeting. Here is a more complete list. When you're done addressing all of them, then submit a final "production ready" version of the `product-digit-sum.cpp` program.

- The types (and modifiers) used should be as *small as possible* but also as *large as necessary*.
- Your program should reject user inputs for a and b that are negative integers.
- Your program should have a function `pow(a, b)` that computes a^b. You should not use any math modules.
- Your program should have a function `vectorize_digits(n)` that takes a single number and creates a vector of the digits in that number.
- Your program should have a function `sum_vector(v)` which takes a vector of numbers and returns the sum of those numbers.
- Your program should have a function `vec_to_string(v)` which takes a vector of numbers and returns a string representation. This function is just used for debugging purposes.
- Your program should have a function `tests()` that runs *fully automated* tests (using `assert` statements) for all of these functions.
- Your program should have a main method that uses the other methods you've built (except for `test()`) to actually solve the problem.
- Your program should have a Makefile.
- Your program should return appropriate exit codes.

After making the necessary refactoring to your program, make a new commit using git and push the changes to your existing repository on github.

Because of this modular design (many small functions) we can re-use some of this code to compose a different program `split-list-balance.cpp`. You are given a vector of numbers `list` and a number `M`. The numbers in `list`, as well as `M`, must be positive and non-zero.

```
list = {2, 3, 4, 5, 6}        M = 2
```

Your task is to break `list` into `M` continuous, non-empty sub-lists such that the largest `sum_vector(sub-list)` for all the sub-lists is minimal. For the above example the solution is 11, wherein `list` is split into `{2, 3, 4}` and `{5, 6}`. `sum_vector({5, 6}) = 11`.

You should implement your solution in a function:

```
int split_vec(vector<int> vec, int m);
```

Your main function should contain any / all tests of `split_vec()` as you see fit.

Notes

- This problem is hard! Work it out on paper / by hand first. Draw pictures and think.
- Multiple possible solutions may exist. You only need to find any one of them.

- Your program should output the sum and not the sub-lists themselves. Consider the sample output below.
- In the example input the numbers are sorted. That is not always the case.
- Please re-use any of the functions from the product digit sum program in this program. For example, for debugging purposes it will probably be very useful to convert vectors of ints to a string for printing: `string vec_to_string(vector<int> vec);`

Sample Output:

```
user@machine:solution$ ./slb
input: [1, 4, 4]  m: 3
ans: 4

user@machine:solution$ ./slb
input: [7, 2, 5, 10, 8]  m: 2
ans: 18

user@machine:solution$ ./slb
input: [2, 3, 4, 5, 6]  m: 2
ans: 11
```

You can / should hard-code the input vector and `M` into your program. But be sure to test different inputs to make sure your program works (I will when grading!). Submit your program as `split-list-balance.cpp` in the same repo as your revised `power-digit-sum.cpp` Your `Makefile` should compile both programs in the `all` directive. The addition of this program should be at least one more commit in the same "HW2" repo.

Submission: Your program will be submitted using canvas and git + github.

1) You should already have a private "HW2" repo with `fmresearchnovak` as a collaborator.

2) Make the necessary commit(s) to the repo before the submission deadline for part 2.

3) On canvas you should upload a link to this github repository
`https://github.com/fmresearchnovak/HW2.git`

I will grade the final commit made before the due-date. This means that you can actually submit on canvas at any time before the deadline, and still make changes (new commits) to the code.