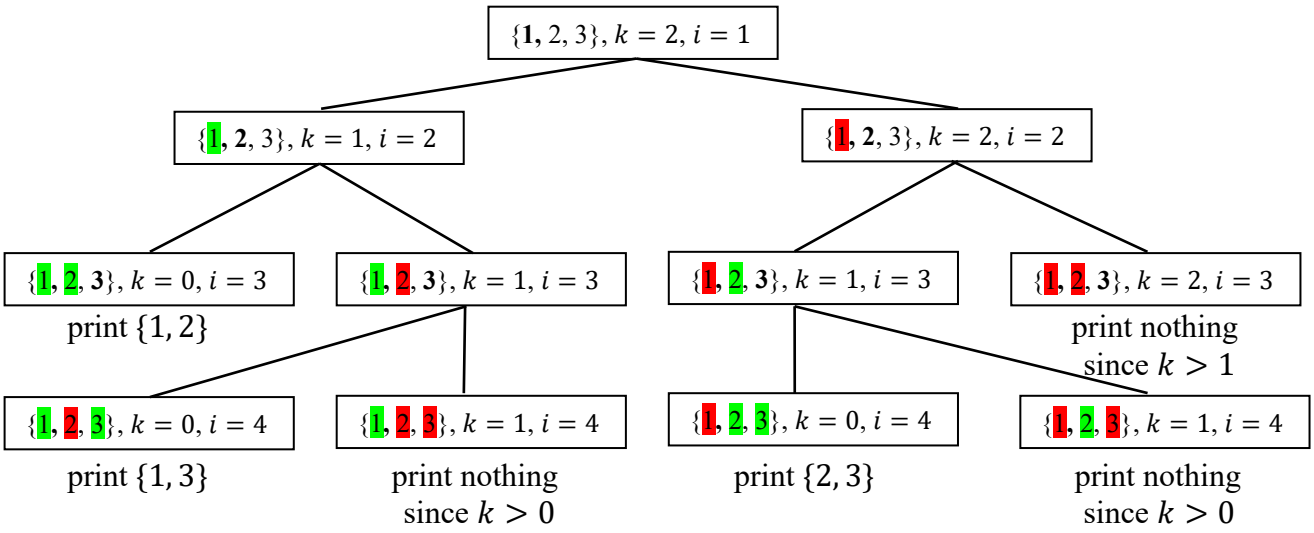
CSE 30
Programming Abstractions: Python
Programming Assignment 1

In this assignment you will write a Python program that uses recursion to print out all k-element subsets of the n-element set $\{1, 2, 3, \ldots, n\}$, where both n and k are given on the command line. Recall that the Binomial Coefficient $C(n, k)$ is defined to be the number of such subsets, and that these numbers can be computed recursively using Pascal's identity. For instance, if $n = 4$ and $k = 2$, one verifies $C(4, 2) = 6$, and that the 2-element subsets of $\{1, 2, 3, 4\}$ are: $\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}$. The core idea of our recursive solution is essentially the same as in the proof of Pascal's identity.

$$C(n,k) = \begin{cases} 1 & \text{if} \quad k = 0 \text{ or } k = n \\ C(n-1, k-1) + C(n-1, k) & \text{if} \quad 0 < k < n \end{cases}$$

At each level of the recursion, we consider a particular element i in the set $\{1, 2, 3, \ldots, n\}$. First construct all k-element subsets that include i, and then construct those subsets that exclude i. The following box trace illustrates the case $n = 3$, $k = 2$.



Observe that the element i under consideration at each level is in **bold**. Also green highlight indicates that an element has been included in the subset, and red highlight indicates exclusion. For instance, at the top level, the element 1 is being considered. The left branch includes 1, and the right branch excludes it. The variable k always indicates the number of elements yet to be selected, so at each left branch k is decremented, while at each right branch it stays the same. When we reach a box in which $k = 0$, no more elements need to be selected, so we print the chosen (green) elements. These form a completed subset of the required size. At each box, the number of elements whose inclusion is undetermined is $n - i + 1$. If we reach a box in which k is greater than this number, i.e. $k > n - i + 1$, we return without printing anything, since that box cannot generate a subset of size k.

As an exercise, construct a box trace for the case $n = 4$, $k = 2$, obtaining the six 2-element subsets listed above. (Be sure to use a large piece of paper when you do this.) It is highly recommended that you complete several such traces by hand before you attempt to write any code for this project. In general, you cannot instruct a computer to do something that you cannot do yourself.

Representation of Subsets

We will use the following well-known approach to represent subsets of $\{1, 2, 3 \ldots, n\}$. First initialize list B of length $n + 1$, and fill it with 0's and 1's. For any index i in the range $1 \leq i \leq n$, set

$$B[i] = \begin{cases} 1 & \text{iff } i \text{ is included} \\ 0 & \text{iff } i \text{ is excluded} \end{cases}$$

The list element $B[0]$ will not be used. Thus the bit-list $B = [*, 0, 0, 1, 0, 1, 1, 0]$ represents the subset $\{3, 5, 6\}$ of $\{1, 2, 3, 4, 5, 6, 7\}$. Here $*$ represents whatever you choose to put in $B[0]$. As an exercise, re-do the above box trace (and any other box traces you have performed) representing each subset as a bit list. Note this exercise is essential to success in this assignment, don't fail to do it.

Some students may dislike the idea of throwing away $B[0]$. It is possible to write the required functions for this assignment without using this convention, but then those functions would not be compatible with the tests used to evaluate your program. To get full credit, you must represent subsets of $\{1, 2, 3, \ldots, n\}$ with an bit list of length $n + 1$ containing only 0's and 1's in $B[1], B[2], B[3], \ldots, B[n]$. You can put something in $B[0]$ if you like, but it will not be examined during grading of your project.

There are two required functions in this assignment. A function with the heading

```
def to_string(B):
```

will be included that converts a bit list to the corresponding string representation of a set using braces ("`{`" and "`}`"), commas and positive integers. Thus if B is in the state $[*, 0, 0, 1, 0, 1, 1, 0]$, then the string

```
'{3, 5, 6}'
```

will be returned. Note the empty set will be represented as a list of all 0's, and the above function will return the string '`{ }`'. (That's a pair of open and closing braces with a single space between.) Your program will include another function with heading

```
def printSubsets(B, k, i):
```

That implements the recursive algorithm we've been discussing. The main part of your program will be placed after a conditional statement.

```
if __name__=='__main__':
```

All function definitions must come before this statement. Also, no variables may be defined before it. This is again to assure your program can be treated as a module, to facilitate grading. After the the above conditional, your program will read command line arguments, initialize a list of appropriate size, then call the required functions to accomplish the tasks described below. You may of course include other helper functions as you see fit. See `Examples/pa1/Args.py` to see how to read command line arguments.

Program Operation

Your program will be called `Subset.py` and will operate at the Unix command line as follows.

```
$ python3 Subset.py 4 2
{1, 2}
{1, 3}
{1, 4}
{2, 3}
{2, 4}
{3, 4}
$ python3 Subset.py foo bar
$ python3 Subset.py 6
$ python3 Subset.py 6 7
$ python3 Subset.py 8 0
{ }
$ python3 Subset.py 5 3
{1, 2, 3}
{1, 2, 4}
{1, 2, 5}
{1, 3, 4}
{1, 3, 5}
{1, 4, 5}
{2, 3, 4}
{2, 3, 5}
{2, 4, 5}
{3, 4, 5}
$ python3 Subset.py 3 3
{1, 2, 3}
$ python3 Subset.py 3 0
{ }
```

As you can see, if anything other than two command line arguments are supplied, or if those two arguments cannot be parsed as ints, or if they can be parsed as ints, but do not satisfy the inequality $0 \leq k \leq n$, then nothing will be printed, and the program will terminate. Each subset is to be printed on a single line of output. Your program must match the above format exactly to receive full credit.

What to turn in

Submit the file `Subset.py` to the assignment pa1 on Gradescope before the due date. Some sample output has also been posted in `Examples/pa1` on the class webpage.