

# Computational Project — Exhaustive Enumeration in Counting Classic

Generate and count combinations/arrays from  $X = \{1, \dots, n\}$

(Write the group names here)

(delivery date)

Document structure. This file contains two parts:

1. The problems (what the student must implement and report).
2. The solution (expected answers and output examples).

## PART I — The problems (different project: 100% by enumeration)

In this project, closed formulas are not used for the main computation. Instead, the programs must exhaustively generate combinatorial structures from the standard set  $X = \{1, \dots, n\}$ , print each one, and count them to obtain the value of each function. The results can then be compared against the formula only for verification.

### Operations to be implemented (all by enumeration and printing)

For each point (a)–(f), write a function that:

- generate all instances from  $X$ ,
- print them (one per line or separated by spaces),
- and finally print the total count found.

(a) Permutations of  $X$  (all elements). List all permutations of length  $n$  of  $X$ ; print each permutation; at the end print the total (it should match  $n!$  upon verification).

(b)  $r$ -permutations without repetition. Given  $n, r$  ( $0 \leq r \leq n$ ), enumerate all tuples of length  $r$  with distinct elements of  $X$ ; print them and count (then check against  $P(n, r)$ ).

(c) Combinations without repetition. Given  $n$  and  $r$ , list all subsets of size  $r$  of  $X$ ; print them (in ascending order) and count (then check against  $\binom{n}{r}$ ).

(d)  $r$ -permutations with repetition. Given  $n, r$ , enumerate all  $r$ -tuples with replacement taken from  $X$  (Cartesian product); print and count (then check against  $n^r$ ).

(e) r-combinations with repetition (multisets). Given  $n$  and  $r$ , enumerate all  $r$ -selections of  $X$  regardless of order but allowing repetitions (non-decreasing lists); print and count (then check against  $\binom{n+r-1}{r}$ ).

(f) Stars and Bars. Given  $m, k$ , enumerate all  $k$ -tuples  $(x_1, \dots, x_k)$  of integers  $\geq 0$   $\sum x_i = m$ ; print each  $k$ -tuple with and count (then check against  $\binom{m+k-1}{k-1}$ ).

## Suggested minimum interface

- A single file, `EnumerationCount.py`, that receives command-line arguments:

# examples:

```
python CountEnumeration.py --op perm --n 4
pythonCountEnumeration.py --op nPr --n 5 --r 3
pythonCountEnumeration.py --op nCr --n 6 --r 2
pythonCountEnumeration.py --op nPr_rep --n 3 --r 4
pythonCountEnumeration.py --op nCr_rep --n 4 --r 3 python
CountEnumeration.py --op bars --m 5 --k 3
```

- Optional flag – verify that, after counting by enumeration, it compares against the known formula and prints OK/FAIL.
- Mandatory printing of all generated combinations/arrays. To avoid huge outputs, parameters will be restricted (see next section).

## Restrictions and error handling

- To ensure that the entire printout is feasible, use safe limits by default (modifiable via command line):

$n \leq 7, r \leq 6, m \leq 10, k \leq 6$ .

- If the parameters exceed the threshold, the program must reject execution with a clear message (or request –force to allow it).
- Validate  $0 \leq r \leq n$  when there is no repetition. Negative parameters – error.

## Boundary exploration (required, brief)

Implement a subcommand –limits that measures the enumeration time (not the formulas) and reports the largest size that allows printing everything in less than  $T = 2$  s for each operation.

Suggested format:

```
[Limits] perm: n<=7 (1.3s) ; n=8 (3.0s) too slow.
[Limits] nPr: n<=7,r<=5 (1.8s) ; r=6 (2.7s) slow.
...
```

Add a brief commentary line interpreting the growth in cases.

## Evaluation criteria (20 pts)

1. Correctness of the enumeration and complete printing (8 pts).
  2. Interface and error handling (4 pts).
  3. Exploration of limits: simple design, clear times and thresholds (4 pts).
  4. Optional verification against formulas and clarity of output (2 pts).
  5. Style and organization of the code (2 pts).
-