\* 1. Je dána množina malých písmen P = {'a', 'b', 'c', ..., 'z'} bez diakritiky. Máme vygenerovat tabulku T s 8 sloupci, jejíž každý řádek bude obsahovat jednu 8 prvkovou podmnožinu množiny P tak, že každá buňka na řádku bude obsahovat právě jeden znak. Tabulka bude obsahovat všechny možné navzájem různé 8 prvkové podmnožiny P a žádná podmnožina se v T nebude opakovat. Určete, jak bude tabulka velká a zda ji váš počítač bude moci vyplnit během 1 sec.

\* 2. Napište pseudokód funkce, která vypíše všechny neprázdné podmnožiny množiny {0, 1, 2, ..., *n*─1}.

\* 3. Mějme permutace množiny M = {1, 2, 3, ..., *n*}, *n* > 4. Permutaci *p* této množiny prohlásíme za *přívětivou*, pokud platí: *p*(3) ∈ {3, *n*},  *p*(*n*) ∈ {3, *n*}, *p*(1) = 1, *p*(2) = 2, p(*i*) ∈ {4, ..., *n*−1} pro *i* = 4, ..., *n*−1.

Určete počet *přívětivých* permutací množiny M.

\* 4. Předpokládejme, že každý prvek Gray code G*n*, jímž je *n*-tice nul a jedniček, bude uložen v poli znaků o délce *n*. Napište pseudokód rekurzivní funkce, která pro dané *n* vygeneruje a vypíše celý Gray code G*n*.

\* 5. Všechny permutace množiny M s 98 prvky očíslujeme pořadovými čísly od 0 do 98!−1. V programu pak nepracujeme s permutacemi ale jen s jejich pořadovými čísly. Víme, že budeme zkoumat pokaždé najednou 100 permutací, čili v paměti budeme muset mít uloženo právě 100 pořadových čísel různých permutací množiny M. Kolik minimálně bitů si musíme v paměti rezervovat, abychom si těchto 100 reprezentací mohli uložit?

6. Uvažujme všechny *k*-prvkové podmožiny množiny M = {1, 2, 3, ..., *n*}, 1 ≤ *k* ≤ *n.* Vyjděte z algoritmu transformujícího seznam prvků jedné podmnožiny na seznam prvků podmožiny bezprostředně následující v lexikografickém uspořádání těchto podmnožin. Navrhněte a popište algoritmus, který bude transformovat seznam prvků jedné podmnožiny na seznam prvků podmožiny bezprostředně předcházející v témže lexokografickém uspořádání. Bude mít stejnou asymptotickou složitost?

7. Uvažujeme permutace množiny M = {1, 2, 3, ..., *n*}. Cyklus délky *k* v permutaci *p* definujeme jako množinu

A = {*a*1, *a*2, ..., *ak*} ⊆ M, pro kterou platí:

1 ≤ *a*1 < *a*2 < ... < *ak* ≤ *n*, p(*aj*) = *aj*+1 pro 1 ≤ *j* < *k*, *p*(*ak*) = *a*1.

Určete, kolik je takových permutací množiny {1, 2, 3, ..., *n*}, které obsahují právě dva cykly, z nichž jeden má délku 4 a druhý délku *n*─4.

8. Rank permutace π množiny N = {0, 1, 2, ..., *n*─1} je pořadové číslo této permutace v seznamu všech permutací množiny N uspořádaném v rostoucím lexikografickém pořadí, přičemž prvky seznamu jsou číslovány od 0. Napište pseudokód funkce která v čase úměrném *n* vytiskne takovou permutaci π množiny N, jejíž rank je právě *n*!/2. Předpokládáme *n* ≥ 2.

9. Uvažujme všechny permutace množiny M = {1, 2, 3, ..., *n*}*.* Vyjděte z algoritmu transformujícího danou permutaci na permutaci bezprostředně následující v lexikografickém uspořádání. Navrhněte a popište algoritmus, který bude transformovat danou permutaci na permutaci bezprostředně předcházející v témže lexokografickém uspořádání. Bude mít stejnou asymptotickou složitost?

10. Permutace *p* množiny {1, 2, 3, ..., *n*} se nazývá derangement (rusky беспорядок = nepořádek), pokud platí

1 ≤ *k* ≤ *n*  ⇒ *p*(*k*) ≠ *k.* Napište všechny derangement-y množiny{1, 2, 3, 4}.

Určete 1000000-tý prvek v lexikografickém uspořádání derangement-ů množiny {1, 2, 3, ..., 20}.

11. Osmiprvková posloupnost P = (000, 001, 011, 110, 111, 101, 100) představuje Gray code G3. Dvě konečné posloupnosti A a B prohlásíme za ekvivalentní, pokud:

1. Obrácením pořadí prvků v A získáme posloupnost B nebo

2. Rotací o 1 nebo více pozic doleva nebo doprava posloupnosti A získáme posloupnost B nebo

3. Existuje posloupnost C ekvivalentní s A i s B.

Najděte 8 prvkovou posloupnost Q představující Gray code, která není ekvivalentní s P. Pozor, Gray code je každá binární soustava, v níž se sousední prvky liší právě jedním bitem.