Algoritmos e Estruturas de Dados II
EC3

Nome: Lucas Gonçalves Santos
Matrícula: 202120303511

Questão 1. Implementar (entregar o programa e a saída impressa) um programa de Backtracking para gerar as 100 primeiras configurações para:
AR(100,4) = arranjos de 100 elementos com repetição, 4 a 4.

```python
def arranjoComRepeticao(v, p, cont):
    if cont >= 100:
        return cont
    if (len(p) == 4):      #combinação 4 a 4
        print(p, file=arq)
        cont = cont + 1
        return cont
    for i in range(len(v)):
            p.append(v[i])
            cont = arranjoComRepeticao(v, p, cont)
            p.pop()
    return cont

arq = open("arranjoRep.txt", "w")
v = list(range(1,101))
p = []
cont = 0
arranjoComRepeticao(v, p, cont)
arq.close()
```

PC(100) = permutações circulares de 100 elementos.

```python
def permCirc(vetor, p, usado, cont):
    if cont >= 100:
        return cont
    if len(p) == (len(vetor) - 1):
        p.append(aux)
        print(p,file=arq)
        p.pop()
        cont = cont + 1
        return cont
    for i in range(len(vetor) - 1):
        if usado[i] == False:
            usado[i] = True
            p.append(vetor[i])
            cont = permCirc(vetor, p, usado, cont)
            usado[i] = False
            p.pop()
    return cont

arq = open("permCircular.txt", "w")
vetor = list(range(1,101))
aux = vetor[len(vetor) - 1]
p = []
usado = [False] * len(vetor)
cont = 0
permCirc(vetor, p, usado, cont)
arq.close()
```

C(100, 5) = combinações de 100 elementos, 5 a 5.

```python
def combinacao(np, contador):
    for i in range(1, n + 1):
        if contador == 100:
            return contador
        if i > p[np - 1]:
            p[np] = i
            if np == q:
                print(p[1:], file = arquivo)
                contador = contador +1
                if(contador == 100):
                    return contador
            else:
                contador = combinacao(np + 1, contador)
                if contador == 100:
                    return contador
    return contador

arquivo = open("combinacao.txt", "w")
contador = 0
n = 100  #elementos
q = 5    # 5 a 5
p = [-1] * 6
p[0] = 0
combinacao(1, contador)
arquivo.close()
```

Questão 2. Modifique o mínimo possível o algoritmo de backtracking mostrado a seguir para geração de subconjuntos de um conjunto V de números inteiros, tal que o número de elementos ímpares nos subconjuntos impressos seja o dobro do número de elementos pares.

```
GeraSub (ns, t):
   para i ← t..n incl.:
     S[ns] ← V[i]
     para j <- 1..ns(inclusive):
          se (S[j] MOD 2 == 0):
               numPares++
          senao:
               numImpares++
          se(numImpares == (2*numPares)):
               escrever(s)
     se (i<n)
        GeraSub (ns+1, i + 1)

para i ← 1..n incl.:
   ler V[i]
GeraSub (1, 1)
```