## Exercício 1

- A - As funções de hash utilizadas são a MurmurHash2, a SHA256, SHA512 e a MD5.
- B - O projeto foi realizado na plataforma IntelliJ IDEA, quando corre o projeto, após selecionar a opção de algoritmo, deve conseguir escolher o valor de $n$, para gerar $2^n$ transações.
- C - O output está conforme o exemplo que o processor incluiu no ficheiro do exercício. O exemplo abaixo exemplifica um output, para o algoritmo MD5 e 8 transações.

```
Choose a hashing algorithm:
1. SHA-256
2. SHA-512
3. MD5
4. MurmurHash3
Enter algorithm choice (1-4): 3

Choose a number n to raise to the power:
3
Initialize
Num. Transaction levels 3  transactions 8

Merkle Tree:
                              e3f7
              36bc                              14a7
      28b1             30c7             6285             1d8b
  ffec    d8da    6963    aa4d    72af    1a1e    bfd0    2523

Proof for: tx1
tx1 [ d8daaf40 30c7249c 14a7a6d7 ] e3f76992 Validated
tx1 [ 00000000 30c7249c 14a7a6d7 ] e3f76992 False

Process finished with exit code 0
```

- D - E - Como se pode verificar no exemplo de output, as provas são feitas à root.
- F - **Para** $2^{10}$, existem 1024 transações no total - O tempo necessário para validar uma transição válida é de 1,2 ms (milissegundos), e o tempo para validar uma transição falsa foi igual.
- F - **Para** $2^{20}$, existem 1048576 transações no total - O tempo necessário para validar uma transição válida é de 17,4 ms (milissegundos), e o tempo para validar uma transição falsa foi de 20,3 ms.
- F - **Para ambos os casos** - O brute force seria muito mais lento, pelo que

uma só prova é mais simples de provar, do que computar a merkle tree novamente, e passar por todas as transações. E, seria possível acelerar estas provas, se utiliza-se trees em cache, ou seja valores de hash intermediários, aquando a verificação de várias transações.