Note 05: Runtimes

Instruction

For each function below, for its worst-case scenario,

- Construct its runtime table.
- Calculate its runtime function.
- State and prove its theta-notation by providing the values of c_1, c_2, and n_1 from the definition of theta-notation.

Functions:

```
01 long A(long x)
02 {
03   if(x < 0)
04   {
05     x *= -1;
06   }
07
08   long n = 0;
09
10   for(;x > 0;n += 1)
11   {
12     x = x / 10;
13   }
14   return n;
15 }
```

```
01 template <typename T>
02 long B(const Array<T>& data,const T& value)
03 {
04   long cnt;
05
06   for(long i = 0;i < data.Length();i += 1)
07   {
08     if(data[i] == value)
09     {
10       cnt += 1;
11     }
12   }
13   return cnt;
14 }
```

```
01 template <typename T>
02 bool C(const Array<T>& data)
03 {
04   for(long i = 0;i < data.Length();i += 1)
05   {
06     for(long j = i + 1;j < data.Length();j += 1)
07     {
08       if(data[i] == data[j])
09       {
10         return true;
11       }
12     }
13   }
14   return false;
15 }
```

```
01 string D(int n)
02 {
03   if(n < -1000 || n > 1000)
04   {
05     return "not sure";
06   }
07   else if(n < 0)
08   {
09     n *= -1;
10   }
11   Array<int> p = {2,3,5,7,11,13,17,19,23,29,31};
12
13   for(long i = p.Length() - 1;i >= 0;i -= 1)
14   {
15     if(n % p[i] == 0)
16     {
17       return to_string(p[i]);
18     }
19   }
20   return to_string(n);
21 }
```

```
01 template <typename T>
02 long E(const Array<T>& data,const T& value)
03 {
04   for(long i = data.Length() - 1;i >= 0;i -= 1)
05   {
06     if(data[i] == value)
07     {
08       return i;
09     }
10   }
11   return data.Length();
12 }
```

```
01 bool F(const Array<bool>& data)
02 {
03   long n = data.Length() - 1;
04
05   for(long i = 0;i < (n + 1) / 2;i += 1)
06   {
07     if(data[i] != data[n - i])
08     {
09       return false;
10     }
11   }
12   return false;
13 }
```