Exercise 1: Distributions

(a) For each of the distributions, come up with at least two different methods of drawing random variables from it using a sequence of uniform i.i.d. random variables X_i.

   (i) Geometric distribution with parameter p: The Geometric distribution has CDF $F(x) = 1 - (1-p)^{x+1}$ for $x \in \{0, 1, \dots\}$. It describes the number of failures before the first success in a series of independent Bernoulli trials with success probability p.

   (ii) Cauchy distribution with location parameter μ and scale parameter b: The Cauchy distribution has CDF $F(x) = \frac{1}{2} + \frac{1}{\pi}\arctan(\frac{x-\mu}{b})$ for $x \in \mathbb{R}$. If (X, Y) are the 2D coordinates of a point uniformly distributed inside the unit circle, then X/Y is Cauchy($\mu = 0, b = 1$)-distributed.

   (iii) Poisson distribution with parameter λ: The Poisson distribution has PMF $P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$. It counts the number of events in a unit time interval if the time between events is exponentially distributed with parameter λ.

(b) If X an Y are independent standard normal random variables, what is the distribution of $X^2 + Y^2$? Using this result, write a function that draws normal random variables using uniform random variables and inversion sampling *without* requiring the probit function.

Exercise 2: Mersenne Twister

Write a function `mersenneStep` that implements one step of the Mersenne Twister in R, using operations from the `bitops` package for bitwise operations (see the package's documentation). You do not need to implement the initialization of the state vector.

The function should take a (624-dimensional) state vector `x` as input, and return a list with the elements `x` and `u`. The `x` element of the returned list should be the updated state vector, and the `u` element should be a vector of length 624 with the uniform random variables.

Note, because the Mersenne Twister generates 32-bit integers, you need to convert them to $(0, 1)$-uniform random variables by dividing by 2^{32}.

Your function should match the behaviour of the built-in `runif` function exactly, meaning that when given the RNG state from `.Random.seed`, it should produce the same sequence of uniform random variables. The expected behaviour is as follows:

```r
set.seed(1)
x <- .Random.seed[-c(1, 2)]  # the first two elements are meta-information

mstep <- mersenneStep(x)
unifs <- runif(624)
identical(unifs, mstep$u)
#> [1] TRUE

mstep <- mersenneStep(mstep$x)
unifs <- runif(624)
identical(unifs, mstep$u)
#> [1] TRUE
```

You find the algorithm for the Mersenne Twister described in the lecture slides. The constants used by R are:

Description	Constant	Value
Word size	w	32
Words in state vector	n	624
Offset for recurrence relation	m	397
Bit position for concatenation	c_{sep}	31
Coefficient of the twist matrix	a	`0x9908B0DF`
Shift constant	u	11
Shift constant	s	7
Shift constant	t	15
Shift constant	l	18
Mask constant	d	`0xFFFFFFFF`
Mask constant	b	`0x9D2C5680`
Mask constant	c	`0xEFC60000`

Note, however, that R reserves the integer value `0x80000000` for `NA`.

```
bitwShiftL(0x40000000, 1)

#> [1] NA
```

You should therefore not use `integer` vectors, but `numeric` vectors and the `bitops` package. However, since R uses `integer` vectors internally, this means that there are rare cases where the `.Random.seed` vector will contain `NA`s. Since this is only an exercise, you can ignore this.