Напишите функцию `download`. Функция `download`:

- на вход получает адреса для скачивания - `urls`
- конкурентно скачивает информацию из каждого `url` (для скачивания используйте функцию `fakeDownload`)
- если вызовы `fakeDownload` возвращают ошибки, то нужно вернуть их все (см. errors.Join)

```
package main

import (
 "errors"
 "fmt"
 "math/rand"
 "sync"
 "time"
)
// timeoutLimit - вероятность, с которой не будет возвращаться ошибка от fakeDownload():
//  timeoutLimit = 100 - ошибок не будет;
//  timeoutLImit = 0 - всегда будет возвращаться ошибка.
// Можете "поиграть" с этим параметром, для проверки случаев с возвращением ошибки.
const timeoutLimit = 90

type Result struct {
 msg string
 err error
}

// fakeDownload - имитирует разное время скачивания для разных адресов
func fakeDownload(url string) Result {
 r := rand.Intn(100)
 time.Sleep(time.Duration(r) * time.Millisecond)
 if r > timeoutLimit {
  return Result{
   err: errors.New(fmt.Sprintf("failed to download data from %s: timeout", url)),
  }
 }

 return Result{
  msg: fmt.Sprintf("downloaded data from %s\n", url),
 }
}
// download - параллельно скачивает данные из urls
func download(urls []string) ([]string, error) {
 // напишите ваш код здесь

}

func main() {
 msgs, err := download([]string{
  "https://example.com/e25e26d3-6aa3-4d79-9ab4-fc9b71103a8c.xml",
  "https://example.com/a601590e-31c1-424a-8ccc-decf5b35c0f6.xml",
  "https://example.com/1cf0dd69-a3e5-4682-84e3-dfe22ca771f4.xml",
  "https://example.com/ceb566f2-a234-4cb8-9466-4a26f1363aa8.xml",
  "https://example.com/b6ed16d7-cb3d-4cba-b81a-01a789d3a914.xml",
 })

 if err != nil {
  panic(err)
 }

 fmt.Println(msgs)
}
```