gofuzz

gofuzz is a library for populating go objects with random values.

GO reference build failing

This is useful for testing:

- Do your project's objects really serialize/unserialize correctly in all cases?
- Is there an incorrectly formatted object that will cause your project to panic?

Import with `import "github.com/google/gofuzz"`

You can use it on single variables:

```
f := fuzz.New()
var myInt int
f.Fuzz(&myInt) // myInt gets a random value.
```

You can use it on maps:

```
f := fuzz.New().NilChance(0).NumElements(1, 1)
var myMap map[ComplexKeyType]string
f.Fuzz(&myMap) // myMap will have exactly one element.
```

Customize the chance of getting a nil pointer:

```
f := fuzz.New().NilChance(.5)
var fancyStruct struct {
  A, B, C, D *string
}
f.Fuzz(&fancyStruct) // About half the pointers should be set.
```

You can even customize the randomization completely if needed:

```
type MyEnum string
const (
        A MyEnum = "A"
        B MyEnum = "B"
)
type MyInfo struct {
        Type MyEnum
        AInfo *string
        BInfo *string
}

f := fuzz.New().NilChance(0).Funcs(
        func(e *MyInfo, c fuzz.Continue) {
                switch c.Intn(2) {
```

```
                case 0:
                        e.Type = A
                        c.Fuzz(&e.AInfo)
                case 1:
                        e.Type = B
                        c.Fuzz(&e.BInfo)
                }
        },
)

var myObject MyInfo
f.Fuzz(&myObject) // Type will correspond to whether A or B info is set.
```

See more examples in `example_test.go`.

Happy testing!