Question	Answer 1	Answer 2	Answer 3	Answer 4			

What does the following code print?
 	
`package main`
 	
`import (`
`"fmt"`
`"math"`
`)`
 	
`func main() {`
`fmt.Println(math.NaN() == math.NaN())`
`fmt.Println(math.Inf(1) == math.Inf(1))`
`}`
	true
	true
		false
		true
			true
			false
				false
				false
						https://go.dev/play/p/IAb6hXooFiS	

What does the following code print?
 	
`x := []int{1,2,3,4}`
`x = append(x[:2], x[:copy(x[:2], x[2:])]...)`
`fmt.Println(x)`
	1 2 3 4	1 2 1 2	3 4 3 4	3 4 1 2			

What does the following code print?
 	
`type A struct{}`
 	
`func (a A) String() string {`
`return "A"`
`}`
 	
`type B struct{}`
 	
`func (b B) GoString() string {`
`return "B"`
`}`
 	
`func main() {`
`fmt.Printf("%#v\n", A{})`
`fmt.Printf("%#v\n", B{})`
`}`
	main.A{}
	B
		main.A{}
		main.B{}
			A
			B
				A
				main.B{}
						https://golang.org/pkg/fmt/#GoStringer	

What does the following code print?
 	
`package main`
 	
`import "fmt"`
 	
`type number interface {`
`~int8 | ~int32 | ~int64 | ~int`
`}`
 	
`func add128[T number](x T) T {`
`x = 128`
`return x`
`}`
 	
`func main() {`
`var x int`
`fmt.Println(add128(x))`
`}`
	128	panic	cannot use 128 (untyped i	0		https://go.dev/play/p/fdKJFYsh6zA	

What does the following code print?
 	
`package main`
 	
`import "fmt"`
 	
`func hello(i int) {`
`  fmt.Println(i)`
`}`
 	
`func main() {`
`  i := 5`
`  defer hello(i)`
`  i = i + 10`
`}`
	5	15	error	nil		https://go.dev/play/p/YKes6dtB6AY

What does the following code print?
 	
`package main`
 	
`import "fmt"`
 	
`func main() {`
`  n := 43210 // time in seconds`
`  fmt.Println(n/60*60, "hours and", n%60*60, "seconds")`
`}`
	12 hours and 10 seconds	43200 hours and 600 seconds	43200 hours and 10 secon	12 hours and 600 seconds		

Pythagorean triples are integer solutions to the Pythagorean Theorem, a^2 + b^2 = c^2.
 	
`fmt.Println(3^2+4^2 == 5^2)`
`fmt.Println(6^2+8^2 == 10^2)`
	true
	true
		true
		false
			false
			true
				false
				false		

What does the following code print?
 	
`package main`
 	
`import (`
`  "fmt"`
`  "time"`
`)`
 	
`func main() {`
`  var ch chan int`
`  for i := 0; i < 3; i++ {`
`  go func(idx int) {`
`  ch <- (idx + 1) * 2`
`  }(i)`
`  }`
 	
`  fmt.Println("result:", <-ch)`
`  time.Sleep(2 * time.Second)`
`}`
	0	4	6	fatal error		https://go.dev/play/p/aOd59758BZC
						
						fatal error: all goroutines are asleep - deadlock!

What does the following code print?

```
package main

import (
	"fmt"
	"log"
	"sync"
	"time"
)

func main() {
	defer func() {
		if err := recover(); err != nil {
			log.Println("recovered from panic:", err)
		}
	}()

	var wg sync.WaitGroup
	wg.Add(1)
	go func() {
		defer wg.Done()
		time.Sleep(1 * time.Second)
		panic("panic'ed in goroutine")
	}()

	fmt.Println("waiting for goroutine to finish")
	wg.Wait()
	fmt.Println("finished")
}
```

waiting for goroutine to finish panic: panic'ed in goroutine goroutine 20 [running]: Program exited.	waiting for goroutine to finish panic: panic'ed in goroutine recovered from panic finished	panic: panic'ed in goroutin waiting for goroutine to fini goroutine 20 [running]: Program exited.	panic: panic'ed in goroutine recovered from panic waiting for goroutine to finish finished

https://go.dev/play/p/DMTdsK_5Nd_S

What does the following code print?

```
func main() {
	s := "9"

	v1 := s[0]

	for _, v2 := range s {
		fmt.Println(v1)
		fmt.Println(v2)
		fmt.Println(v1 == v2)
		break // a single loop iteration
	}
}
```

57 9 true	invalid operation	9 9 false	9 9 true

https://go.dev/play/p/cWpsXwBSJ6O

What does the following code print?

```
type S []S

var s S
s = append(s, s)
s = append(s, s)

fmt.Println(s)
```

`[]`	`[[]]`	`[[] []]`	`[[] [[]]]`

What does the following code print?

```
func main() {
	fmt.Println(
		string(
			[]int32{97, 98},
		),
	)
}
```

compiler error	panic	[97 98]	ab

What does the following code print?
 	
`func main() {`
`  s := []int{1, 2}`
`  s = append(s, append(s, s...)...)`
 	
`  fmt.Println(s)`
`}`
	panic: stack overflow	[1 2 1 2]	[1 2 1 2 1 2]	[1 1 1 2 2 2]				

What does the following code print?
 	
`func main() {`
`  s := []int{1, 1, 1, 1}`
`  x := append(s[:2], s[2:]...)`
 	
`  s[2] = 2`
 	
`  fmt.Println(x)`
`}`
	1 1 1 1	1 2 1 2	1 1 2 1	2 2 2 2				

What does the following code print?
 	
`const iota = 5`
 	
`const (`
`  a = iota`
`  b = iota`
`)`
 	
`func main() {`
`  fmt.Println(a, b)`
`}`
	compiler error	0 1	5 5	5 6				