

```

1 //this code is achieved with golang 1.17
2 package main
3
4 import (
5     "fmt"
6     "log"
7     "math/rand"
8     "os"
9     "strconv"
10    "time"
11 )
12
13 //fabLimit should be >=1 for previous checking
14 func outputFabTill(fabLimit int){
15     fabHead :=1
16     fabSecond :=0
17     fmt.Print(fabSecond)
18     for fabHead<fabLimit{
19         fmt.Printf(" %d",fabHead)
20         fabHead+=fabSecond
21         fabSecond=fabHead-fabSecond
22     }
23     fmt.Printf("\n")
24 }
25
26 func main() {
27     //some initialization checking
28     checkStart,err:= strconv.Atoi(os.Args[1]);if err!=nil{
29         log.Fatalf("parsing checkStart(first number failed, int expected
30 ,%v",err)
31     }
32     checkEnd,err:= strconv.Atoi(os.Args[2]);if err!=nil{
33         log.Fatalf("parsing checkEnd(second number) failed, int expected
34 ,%v",err)
35     }
36     numToCheck,err:= strconv.Atoi(os.Args[3]);if err!=nil{
37         log.Fatalf("parsing numToCheck(third number) failed, int expected
38 ,%v",err)
39     }
40     if checkStart<=0||checkEnd<=0||numToCheck<=0{
41         log.Fatalf("0/minus input detected,please check input")
42     }
43     //swap if not in right order
44     if checkStart>checkEnd{
45         checkStart+=checkEnd
46         checkEnd=checkStart
47         checkStart-=checkEnd
48     }
49     //used to save all passed tests
50     passedTests:=[]int{}
51     for

```

```

52     }
53     //failed to pass, exit
54     if len(passedTests)==0{
55         panic("numToCheck failed to pass a single task, exiting")
56     }
57     //output randomly in passed testes
58     rand.Seed(time.Now().UnixNano())
59     outputFabTill(passedTests[rand.Intn(len(passedTests))])
60 }

```

Plain text below

```
package main
```

```
import (
    "fmt"
    "log"
    "math/rand"
    "os"
    "strconv"
    "time"
)
```

```
//fabLimit should be >=1 for previous checking
```

```
func outputFabTill(fabLimit int){
    fabHead :=1
    fabSecond :=0
    fmt.Print(fabSecond)
    for fabHead<fabLimit{
        fmt.Printf(" %d",fabHead)
        fabHead+=fabSecond
        fabSecond=fabHead-fabSecond
    }
    fmt.Printf("\n")
}
```

```
func main() {
    //some initialization checking
    checkStart,err:= strconv.Atoi(os.Args[1]);if err!=nil{
        log.Fatalf("parsing checkStart(first number failed, int expected ,%v",err)
    }
    checkEnd,err:= strconv.Atoi(os.Args[2]);if err!=nil{
        log.Fatalf("parsing checkEnd(second number) failed, int expected ,%v",err)
    }
    numToCheck,err:= strconv.Atoi(os.Args[3]);if err!=nil{
        log.Fatalf("parsing numToCheck(third number) failed, int expected ,%v",err)
    }
    if checkStart<=0 || checkEnd<=0 || numToCheck<=0{
        log.Fatalf("0/minus input detected,please check input")
    }
    //swap if not in right order
    if checkStart>checkEnd{
        checkStart+=checkEnd
    }
}
```

```

    checkEnd=checkStart
    checkStart-=checkEnd
}
//used to save all passed tests
passedTests:=[]int{}
for currentChecking:=checkStart;currentChecking<=checkEnd;currentChecking++){
    if numToCheck%currentChecking==0{
        passedTests=append(passedTests,currentChecking)
    }
}
//failed to pass, exit
if len(passedTests)==0{
    panic("numToCheck failed to pass a single task, exiting")
}
//output randomly in passed testes
rand.Seed(time.Now().UnixNano())
outputFabTill(passedTests[rand.Intn(len(passedTests))])
}

```