Arrays and Slices

Array is a data structure whose memory is allocated sequentially. Memory in sequential form lets memory in use stay loaded into CPU caches for a longer period of time. Using index arithmetic, we can loop through all of the items. Because the length of an array is determined by its type, arrays cannot be resized.

Declaring Arrays

```
var array[5] int
```

Declaration and initialization in one line

```
array := [5] int {1,2,3,4,5}
```

We can also assign values to specific indexes

```
/*Here index 1 and index 2 are assigned values 10 and 20
respectively while other indexes are set to default value 0*/

array:= [5]int {1:10,2:20}
```

Slices are tiny objects that abstract and manipulate an underlying array. They're adaptable in terms of growth and include a built-in function called append that allows us to efficiently increase slices. Its also possible to reduce the size of slice by slicing out a part of underlying memory

```
//Creates slice of string with length and capacity of 5
slice := make([]string,5)

//Creates slice of string with length of 3 and capacity of 5
slice2 := make([]string,3,5)

//Slice literal declaration of length and capacity of 3
slice3 := []string{"a","b","c"}
```

We can slice an existing slice to create a new slice

```
slice := []string{"a","b","c"}

//slice2 will be [b,c]
slice2 := slice[1:3]
```

We can append data to the slice

```
s :=[]int{1,2,3,4,5}
s2:= s[1:3]
s2:= append(s2,6)
```

1- Write a program that scan the element values of an array from user and pass it to a method that separates even , odd elements

Sample Output

```
array[0] = 50
array[1] = 11
array[2] = 744
array[3] = 101
array[4] = 5
50 is an Even Element where in index 1
11 is an odd Element where in index 2
744 is an Even Element where in index 3
101 is an odd Element where in index 4
5 is an odd Element where in index 5
```

2- Solve the previous question using slices

Sample Output

```
array[0] = 5
array[1] = 4
array[2] = 11
array[3] = 222
array[4] = 10
Even Part [4 222 10]
Odd Part[5 11]
```

3- Write a program that asks the user to fill a data of new person
Hint:

- Person is a struct
- Process of entering new person is a loop
  if user enter 1 it make him able to create new person , if he press any key program ends

Sample Output

```
PS C:\GoProjects\src\github.com\golang\example\hello> go run .\hello.go
For adding new Person Enter 1 to end program press any key
1
Enter Person's id :
01
Enter Person's name :
Menna
Enter Person's age :
24
New Person added
For adding new Person Enter 1 to end program press any key
1
Enter Person's id :
02
Enter Person's name :
Hana
Enter Person's age :
25
New Person added
For adding new Person Enter 1 to end program press any key
2
PS C:\GoProjects\src\github.com\golang\example\hello>
```