

# Programmazione

(Vers. D)

## Appello di Gennaio 2020

Cognome ..... Nome .....

Matricola ..... Anno di corso ..... (1, 2, ...)

DSA

Alcune delle seguenti domande riguardano il vostro numero di matricola. Scrivete il vostro numero nel seguente schema, una cifra per ogni spazio:

A	B	C	D	E	F

Nel seguito, useremo le lettere ABC... per riferirci alle corrispondenti cifre del numero di matricola. Ad esempio, "il numero DEF" il numero costituito dalle ultime tre cifre del vostro numero di matricola.

Rispondete alle seguenti domande:

1. Assumete che  $x$  e  $y$  siano variabili **int**, e che il loro valore *prima di ciascuno dei seguenti assegnamenti* sia rispettivamente EF e AE. Assumete inoltre che  $z$  sia una variabile di tipo **int** e di valore  $\text{int}\{AB, BC, CD, DE, EF, FA\}$ . Dite quale sarà il valore di  $x$  e  $y$  *dopo* gli assegnamenti indicati<sup>1</sup>

Assegnamento	x	y
$x += z[y\%6]$		
$x, y = x + y, y + z[x \% 2]$		
$x /= z[y\%6]$		
$x, _ = \text{strconv.Atoi}(\text{strconv.Itoa}(z[y\%6]) + \text{strconv.Itoa}(y))$		
$x, y = y + 1, x - 1$		

<sup>1</sup>Il pacchetto `strconv` contiene la funzione `Itoa(x int) string` che converte l'intero  $x$  in una stringa (la sua rappresentazione in base 10). La funzione `Atoi(s string) (int, error)` effettua la conversione inversa, restituendo eventualmente un errore.

2. Scrivete una funzione  $f$  che dato un intero  $x$ , restituisca la slice contenente i suoi divisori (tranne 1 e  $x$  stesso), in ordine decrescente.

3. Considerate la seguente funzione:

```
func f(x []int, y int) (a int) {  
    for _, c := range x {  
        if c % y == 0 {  
            a++  
        }  
    }  
    return  
}
```

Dite che cosa restituisce la funzione dandole come primo argomento la slice `[]int{A,B,C,D,E,F}` e come secondo argomento i seguenti valori (che dipendono dalle cifre del vostro numero di matricola):

Argomento	Valore restituito
A	
B	
C	
D	
E	
F	

4. La seguente funzione prende due slice di stringhe, diciamo  $s1$  e  $s2$ , crea e restituisce una nuova slice che contiene, alternativamente, un elemento di  $s2$  e un elemento di  $s1$ ; quando una delle due slice si esaurisce, la funzione continua copiando quel che rimane dell'altra. Ad esempio, se le slice sono {"topo", "topi", "rana"} e {"cane", "cene", "katai", "telone", "rosi"} il risultato sarà {"cane", "topo", "cene", "topi", "katai", "rana", "telone", "rosi"}.

Riempite le parti mancanti:

```
func f(s1, s2 ..... ) (res ..... ) {
    n1 := len(s1)
    n2 := len(s2)
    for i := 0; i < n2 ..... i < n1; i++ {
        if ..... {
            .....
        }
        if ..... {
            res = append(res, s1[i])
        }
    }
    return
}
```

5. Considerate la seguente funzione ricorsiva:

```
func f(a int, b int) int {
    if b == 0 {
        return a
    }
    return f(a + 2, b - 1)
}
```

- Che valore restituisce la chiamata f(EFA, FED)? .....
- Che valore restituisce la chiamata f(FA, DEA)? .....

6. Qualcuno ha scritto la seguente funzione, ma purtroppo non è molto leggibile. Supponete di volerla chiamare passando come argomenti i valori EA e BF (dal vostro numero di matricola).

```
func f(x int, y int) (int, int) {
    m1 := x < y
    m2 := x % 10 < y / 10
    if m1 || m2 {
        x += 10
        if m1 && m2 {
            y -= 10
        }
    }
    i := x
    t := 0
    for i < y {
        i += x
        t++
    }
    return i, t
}
```

Considerate le variabili elencate nella tabella e, per ciascuna, dite quale è il suo tipo e il suo valore alla fine della funzione, prima che venga eseguito il **return**.

<b>Variabile</b>	<b>Tipo</b>	<b>Valore</b>
x		
y		
m1		
m2		
i		
t		

7. Nel seguente frammento di codice Go, viene definito un tipo `data` e una funzione che data una slice di date e una data specifica, restituisce tutte le date che cadono nello stesso mese e anno della data indicata, ma che la precedono. Purtroppo la funzione contiene 3 errori. Trovateli, cerchiateli, e ricopiate nella tabella sotto l'espressione (o l'istruzione) errata e quella corretta.

```
type data struct {  
    g, m, a int  
}  
  
func findMin(a []data, x data) (res data) {  
    for d := range a {  
        if d.a == x.a && d.m == x.m && d.g < x.g {  
            append(res, d)  
        }  
    }  
    return  
}
```

1		
2		
3		

8. Scrivete una funzione che date due stringhe  $x$  e  $y$  restituisca una slice contenente tutte le combinazioni costituite da un carattere della seconda stringa e un carattere della prima. Non è importante l'ordine in cui le stringhe compaiono nella slice. Per semplicità assumete che le due stringhe non contengano caratteri ripetuti. Ad esempio, se la prima stringa è "voi" e la seconda è "cavie", la slice conterrà le stringhe "cv", "co", "ci", "av", "ao", "ai", "vv", "vo", "vi", "iv", "io", "ii", "ev", "eo", "ei".



9. Considerate le seguenti funzioni.

```
func f(s string, r rune) bool {
    c := true
    for _, v := range s {
        c = v == r
    }
    return c
}
```

```
func k(s string) rune {
    var v rune
    for _, v = range s {
    }
    return v
}
```

```
func g(s string) bool {
    return f(s, k(s))
}
```

```
func h(s string) rune {
    t := len(s) - 1
    return rune(s[t])
}
```

Siano ora  $A$  e  $B$  due stringhe e  $R$  una runa. Per ciascuna di queste affermazioni, dire se sono sempre vere (V), sempre false (F), oppure a volte vere e a volte false (X).

Affermazione	V/F/X
$f(A + B, R) == f(B, R)$	
se $B$ non è vuota, $k(A + B) == k(B)$	
$k("") == h("")$	
se $A$ è una stringa ASCII e non vuota, $k(A) == h(A)$	
$g(A) == true$	