UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI INFORMATICA

Didattica dell'Informatica
a.a. 2019-20

Esercitazione sulla comprensione di programmi

(nome: esComprensioneProgrammi)

Consegna

A coppie svolgete gli esercizi di comprensione di codice che seguono; dopodiché indicate, per ciascuna domanda, il suo scopo didattico, facendo riferimento anche alla tabella in Figura 1, che illustra i diversi aspetti del processo di comprensione di un programma secondo un modello proposto da C. Schulte, detto *Block Model*.

Per il 28/04/20, sempre lavorando nelle stesse coppie, scegliete un esercizio di scrittura di codice che potreste dare ai vostri (ipotetici) alunni e trasformatelo in esercizio di comprensione, cioè preparate una scheda di lavoro (nome: schedaComprensioneProgrammi) con:

- una soluzione,
- domande di analisi e comprensione del codice che coprano più caselle possibili del *block model*.

Per ciascuna domanda annotate inoltre il suo scopo didattico.

Macro structure	(Understanding the) overall structure of the program text	Understanding the „algorithm" of the program	Understanding the goal/ the purpose of the program (in its context)
Relations	References between blocks, e.g.: method calls, object creation, accessing data...	sequence of method calls „object sequence diagrams"	Understanding how subgoals are related to goals, how function is achieved by subfunctions
Blocks	'Regions of Interests' (ROI) that syntactically or semantically build a unit	Operation of a block, a method, or a ROI (as sequence of statements)	Function of a block, maybe seen as sub-goal
Atoms	Language elements	Operation of a statement	Function of a statement. Goal only understandable in context
	Text surface	**Program execution (data flow and control flow)**	**Functions (as means or as purpose), goals of the program**
Duality	***"Structure"***		***"Function"***

Figura 1: Il *Block Model*

1 Programma brutto

La funzione seguente realizza in modo assai discutibile un calcolo che può essere descritto brevemente.

```
1  int func f(int x) {
2    int a, b, c;
3
4    while ( x > 0 ) {
5      a = x % 10;
6      b = 1 - a%2;
7      c += b;
8      x /= 10;
9    }
10   return c
11 }
```

Analizzate il codice sorgente e rispondete per iscritto alle seguenti domande.

1. Senza eseguire il programma al computer, tracciatene l'esecuzione quando `x` è uguale a 1344.

a	b	c	x

2. Date un nome più significativo alle variabili `a`, `b` e `c`.

3. Riassumete con una frase cosa restituisce la funzione `f`.

2 Cifre

Considerate la seguente porzione di codice.

```
1  int main() {
2     int i, len;
3     char s[80];
4     int f[10] = {0};
5
6     scanf("%s", &s);
7
8     len = strlen(s);
9
10    for (i=0; i<len; i++) {
11       c = s[i];
12       if ( '0' <= c && c <= '9')
13          f[ c - '0' ]++;
14    }
15 }
```

Analizzate il codice sorgente e rispondete per iscritto alle seguenti domande. Se avete dubbi, potete testarlo, eseguendolo su casi di input significativi e modificandolo.

1. Perché si usa un ciclo `for`?
2. A cosa serve l'`if` alla riga 12?
3. Se `c` è una cifra, che cosa indica `c - '0'`?
4. Come descrivereste il valore `f[i]`?
5. Spiegate perché `f` è dichiarata con una lunghezza pari a 10.
6. Date un nome più significativo all'array `f`.
7. Riassumete in una frase cosa fa questa porzione di codice.

3 Divisioni

Considerate la seguente porzione di codice.

```
1   var n, b int
2
3   var c []int
4
5   fmt.Scan(&n, &b)
6
7   if n == 0 {
8     fmt.Print(n)
9   } else {
10
11    for n > 0 {
12      c = append(c, n%b)
13      n = n / b
14    }
15
16    for i := len(c) - 1; i >= 0; i-- {
17      fmt.Print(c[i])
18    }
19  }
```

Analizzate il codice sorgente e rispondete per iscritto alle seguenti domande.

1. Spiegate cosa fa il ciclo nelle righe 11-14.
2. Date dei nomi più significativi alle variabili `n` e `b`.
3. Spiegate cosa fa il ciclo nelle righe 16-18.
4. Date un nome più significativo al programma.
5. Riassumete in una frase cosa fa il programma
6. Se `b` vale 10, descrivete il valore di `c[2]` in relazione all'intero `n`
7. Come è dichiarata `c`? Si potrebbe dichiarare `c` come un array? Quale sarebbe lo svantaggio rispetto alla dichiarazione attuale?

4 Quadrato magico

Considerate la porzione di codice seguente. Il codice serve a costruire un quadrato magico memorizzandolo in un array bidimensionale. Un quadrato magico è una disposizione dei numeri $1, 2, \cdots, n^2$ –con n dispari– tale che in ogni riga, in ogni colonna e nelle due diagonali la somma dei numeri sia la stessa.

```
1  var quadrato = [n][n]int{}
2  var inew, jnew int
3
4  i := 0
5  j := n / 2
6  nn := n * n
7
8  for k := 1; k <= nn; k++ {
9    quadrato[i][j] = k
10
11   if i == 0 {
12     inew = n - 1
13   } else {
14     inew = i - 1
15   }
16
17   if j == n-1 {
18     jnew = 0
19   } else {
20     jnew = j + 1
21   }
22
23   if quadrato[inew][jnew] == 0 {
24     i = inew
25     j = jnew
26   } else {
27     i++
28   }
29
30 }
```

Rispondete alle domande seguenti:

1. Come deve essere dichiarato `n`? Che assunzioni ha senso fare sul valore di `n`?
2. Spiegate come è usata la variabile `k` e cosa rappresenta.
3. Spiegate a parole quale valore viene assegnato a `inew` nel blocco `if-else` nelle righe 11-15
4. Spiegate qual è la funzione del blocco `if-else` nelle righe 23-28
5. Senza eseguire il programma al computer, tracciatene l'esecuzione quando `n` è pari a 5.
6. Verificate la correttezza della vostra risposta al punto precedente, completando il programma con la stampa del quadrato magico.
7. Descrivete a parole la strategia per costruire il quadrato magico che è implementata dalla suddetta porzione di codice. Iniziate con "Si parte mettendo il numero 1 al centro della prima riga ..."