Barem Subiect A (2019)

1. Ce va afișa pe ecran următoarea secvență de cod ? **(3p) [Python 2/3]**		
`print ([i for i in range(1,10) if i & 1 == 0])`		
A) [1,2,3,4]	B) [2,4,6,8]	C) [2,3,4,5]
D) [1,3,5,7,9]	E) [1,2,3,4,5,6,7,8,9]	F) [1]

Răspuns corect: B) [orice alt răspuns nu se punctează / nu se dau punctaje parțiale]. „i & 1 == 0" inseamna numere pare (care nu au primul bit (cel mai putin semnificativ) setat).

2. Ce va afișa pe ecran următoarea secvență de cod ? **(3p) [Python 2/3]**		
`print ("anaaremere".split ("a",2) [2].rsplit("e",2)[0])`		
A) aremere	B) anaare	C) aremere
D) aareme	E) are	F) arem

Răspuns corect: F) [orice alt răspuns nu se punctează / nu se dau punctaje parțiale].

3. Ce va afișa pe ecran următoarea secvență de cod ? **(2p) [Python 2/3]**		
`print ({i%5 for i in range(0,1000) if str(i)[-1] not in ["0","5"]})`		
A) {1,2,3}	B) {0,5}	C) {0,1,2,3,4,5}
D) {0,1,2,3,4}	E) {1,2,3,4}	F) {0}

Răspuns corect: E) [orice alt răspuns nu se punctează / nu se dau punctaje parțiale]. „str(i)[-1] not in ["0","5"] " inseamna ca se iau in calcul doar numere care nu sfarsec cu „0" sau cu „5" – ceea ce inseamna ca restul la 5 nu va putea fi vreodata 0.

4.Explicati ce va afișa pe ecran următoarea secvență de cod ? **(3p) [Python 2/3]**
Expresiile regulate nu contin spatii ci fie forma „\s" pentru match pe caracterul spatiu, respectiv „\x20" pentru caracterul spatiu daca e vorba de o inlocuire.

```
import re
s = "Azi am examenul la python !"
s = re.sub("\s\w{2}\s", "merg", s)
s = re.sub("m(\w\w)g","\x20\\1\x20",s)
print (s)
```

Răspuns corect: „Azi er examenul er python !". Prima substitutie cauta cuvinte formate din doua litere (\w{2}) intre 2 spatii si le inlocuieste cu „merg". O sa faca match pe 2 cazuri – pe „am" si pe „la" si va rezulta „Azimergexamenulmergpython !". A doua substitutie cauta „m" urmat de doua litere si apoi de g, face un grup format din cele doua litere si inlocuieste „m" si „g" cu spatiu (\x20). Orice alt răspuns nu se punctează.

5. Ce va afișa pe ecran următoarea secvență de cod ? **(3p) [Python 2/3]**		
`print(sum(list(map(lambda x: (x*x)%10, range(1,10,2))),5))`		
A) 15	B) 25	C) 26
D) 30	E) 150	F) 140

Răspuns corect: D). Range(1,10,2) → [1,3,5,7,9] → ridicate la patrat ➔ [1,9,25,49,81] ➔ modulo 10 pastreaza ultima cifra ➔ [1,9,5,9,1] ➔ peste care se face o suma ➔ 1+9+5+9+1 = 25 ➔ la care se aduna 5 (ultimul parametru de la sum) ➔ 30. Se puncteaza partial (2 pct) raspunul B) (25 pct).

6. Explicați ce va afișa pe ecran următoarea secvență de cod ? **(4p) [Pythonn 2/3]**

```
import re
print (re.split("[1235]+","412363218"))
```

Se face split dupa orice secvenata format din cifrele 1,2,3 si 5. Mai exact 4**123**6**321**8. **Răspuns corect: 4,6,8 – nu se acorda punctaje partiale.**

7. Ce va afișa pe ecran următoarea secvență de cod ? Justificati. **(4p) [Python 2/3]**

```
class A:
      x = {'A':1,'B':2}
      def  Add(self,k,v): self.x[k] = v
      def  Max(self): return max([i for i in self.x])
a1 = A();a2 = A()
for i in "ABCDEFG":
      if (ord(i) % 2 ==0):
            a1.Add(i,0)
      else:
            a2.Add(i,1)
print (a1.Max() + a2.Max())
```

Pentru ca „x" este definit direct in clasa, atat a1 cat si a2 vor primi o referinta catre „x". Prin urmare nu conteaza ca adaugam elemente in a1 sau in a2, se adauga in acelasi dictionar si in final cele doua variabile (a1 ai a2) vor fi identice ca si continut. Metoda Max din clasa A va lua cel mai mare cheie din dictionar care este „G" (se vede fin for i in „ABCDEFG"). NU ESTE NECESAR SA STITI CODURILE ASCII pentru literele mari – cheile o sa fie A B C D E F si G (este evident care este cea mai mare).
Prin urmare, codul va afisa „GG". [orice alt răspuns nu se punctează / nu se dau punctaje parțiale].

8. Ce va afișa pe ecran următoarea secvență de cod ? Justificati. **(4p) [Python 2/3]**

```
import ctypes

class Test(ctypes.Union):
      _fields_ = [("i",ctypes.c_int),("c",ctypes.c_char)]

A = Test()
A.i = 10
A.c = chr(Test.i.offset+ord('A')+Test.c.offset+Test.i.size+ord(A.c))
print (A.c)
```

Clasa „Test" este un union (deci atat campul „i" cat si campul „c" se gasesc la acelasi offset (0). Pentru ca ocupa acelasi spatiu de memorie, daca A.i = 10, atunci A.c = A.i % 256 = 10.
Prin urmare, A.c = (Test.I.offser = 0) + (ord(‚A') = 65) + (Test.c.offset = 0) + (Test.i.size = 4 (sizeof(int))) + ord(A.c)=10 => A.x = 0+65+0+4+10 = 79 – codul lui O. Se puncteaza si daca se scrie ‚O' si daca se scrie 79. Se puncteaza si raspunsul 69 (‚E' – care ar fi echivalentul pe BIG-ENDIAN – doar daca este acompaniat de o explicatie clara din care sa rezulte ca, calculul s-a facut pe BIG-ENDIAN. [orice alt răspuns nu se punctează / nu se dau punctaje parțiale].

9. Ce se afișează la execuția următorului cod (**Python 2.x**) ? Justificati. **(4p)**

```
import struct
for i in struct.pack("@chcchcicxxxx",'A', 1, 'B', 'C', 2, 'D', 3, 'E'):
      print ord(i),
```

c		h		c	c	h		c				i				c	x	x	x	x
65	0	1	0	66	67	2	0	68	0	0	0	3	0	0	0	69	0	0	0	0

Se puncteaza si daca se da codul ascii, si daca in loc de 65 se scrie ‚A', in loc de 66 se scrie ‚B' s.a.m.d.
[orice alt răspuns nu se punctează / nu se dau punctaje parțiale].