Exercícios Unidade00b:
1.

- O que o código abaixo faz?

```
boolean doidao (char n){
      boolean resp= false;
      int v = (int) c;
      if (v == 65 || v == 69 || v == 73 || v == 79 || v == 85 || v == 97 || v == 101 || v ==105 ||
          v == 111 || v == 117){
            resp = true;
      }
      return resp;
}
```


Obs:

boolean doidao (**char** n)
(char n) deveria ser (char c)

Resposta:

a função ***doidao()*** recebe como parâmetro um caractere que logo em seguida se transforma, por meio de um typecast em sua numeração correspondente. Em seguida ocorre uma verificação deste numero do caráter com outros numerais.

Função retorna ***FALSE*** se numeração do caractere não corresponde as do exemplo, retorna ***TRUE*** se corresponde.

2.

- O que o código abaixo faz?

```
boolean doidao (char n){
      boolean resp= false;
      int v = (int) c;
      if (v == 65 || v == 69 || v == 73 || v == 79 || v == 85 || v == 97 || v == 101 || v ==105 ||
          v == 111 || v == 117){
            resp = true;
      }
      return resp;
}

char toUpper(char c){
      return (c >= 'a' && c <=  'z') ? ((char) (c – 32)) : c ;
}

boolean isVogal (char c){
      c = toUpper(c);
      return (c =='A' || c =='E' ||  c =='I' ||  c =='O' || c =='U');
}
```

Fazendo o isVogal, fica mais fácil ...

Resposta:

- toUpper()
  - Recebe como parâmetro um caráter.

- isVogal()
  - Recebe como parâmetro um caráter.
  - Transforma variável que possui caráter em maiúsculo.
  - Retorna **TRUE** se caractere for igual a {A, E, I, O ou U}.
  - Retorna **FALSE** se caractere não for igual a {A, E, I, O ou U}.

3.

- Um aluno desenvolveu o código abaixo, corrija-o:

```
boolean isConsoante(String s, int n){
    boolean resp= true;
    if (n!=s.length()){
      if (s.charAt(n)<'0' || s.charAt(n)>'9'){
        if (s.charAt(n)=='A' || s.charAt(n)=='E' || s.charAt(n)=='I' || s.charAt(n)=='O' ||
          s.charAt(n)=='U' || s.charAt(n)=='a' || s.charAt(n)=='e' || s.charAt(n)=='i' ||
          s.charAt(n)=='o' || s.charAt(n)=='u'){
          resp= false;
        } else{
          n++;
          resp=isConsoante(s, n);
        }
      } else {
        resp=false;
      }
    }
    return resp;
  }
```

1º passo: Indentação

Resposta:

```
boolean isConsoante(String str, int i){
        boolean resp = true;

        if(n == str.length()){
                resp = true;
        } else if(!isConsoante(str.charAt(i))){
                resp = false;
        } else{
                resp = isConsoante(str, i+1);
        }
        return resp;
} // end isConsoante()
```

4.

- Qual é a sua opinião sobre o código **REAL** abaixo?

```
Unidade recuperarUnidadeComCodigoDeUCI(Unidade unidadeFilha) {
	Unidade retorno = null;

	if (unidadeFilha.getCodUci() != null && !unidadeFilha.getCodUci().isEmpty()) {
		retorno = unidadeFilha;
	} else {
		retorno = unidadeFilha.getUnidadeSuperior();
	}

	while (retorno == null || retorno.getCodUci() == null || retorno.getCodUci().isEmpty()) {
		retorno = retorno.getUnidadeSuperior();
	}

	return retorno;
}
```

Resposta:

1- O nome da função é muito grande, ao invés desse tamanho poderia ter um comentaria explicando o que a função faz.
2- O código da muita volta. poderia haver uma simplificação do código.

5.

- Qual é a diferença entre os dois métodos abaixo?

```
int m1(int i){
	return i--;
}


int m2(int i){
	return --i;
}
```

Resposta:

i-- retornar o valor antes de decrementado.
--i retorna o valor depois de decrementado.

6.

- O que o programa abaixo mostra na tela?

```
byte b = 0; short s = 0; int i = 0; long l = 0;

while (true){
    b++;   s++;   i++;   l++;
    System.out.println(b + " " + s + " " + i + " " + l);
}
```

Resposta:

O programa acima mostra cada variável sendo incrementada +1 a cada loop.
Variável **b** poderá ser incrementada no máximo 2^{8} vezes
Variável **s** poderá ser incrementada no máximo 2^{16} vezes
Variável **i** poderá ser incrementada no máximo 2^{32} vezes
Variável **l** poderá ser incrementada no máximo 2^{64} vezes

7.

- Por que o código abaixo imprime [46 - 11]?

```
int x = 23, y = 23;
x = x << 1;
y = y >> 1;
System.out.println("[" + x + " - " + y + "]");
```

Resposta:

1. $23_{10} = 10111_{2}$
2. 10111_{2} << $101110_{2} = 46_{10}$ (Shift left <<, desloca bits para esquerda adicionando um 0 na posição 0)
3. 10111_{2} >> $01011_{2} = 11_{10}$ (Shift right >>, desloca bits para direita, removendo o bit da posição 0)