

[Painel do utilizador](#) ▶ [As minhas unidades curriculares](#) ▶ [Programação](#) ▶ [Avaliação](#) ▶ [Miniteste 1: Prática \(9/04/2022, 12:00\)](#) ▶

Início sábado, 9 de abril de 2022 às 11:43

Estado Prova submetida

**Data de
submissão:** sábado, 9 de abril de 2022 às 13:13

Tempo gasto 1 hora 30 minutos

Nota 84 do máximo 100



Pergunta 1

Correta Pontuou 20 de 20

Write a C++ program to check whether a triangle is equilateral, isosceles or scalene. An equilateral triangle has all the three sides of the same length, an isosceles triangle has exactly two equal sides, and a scalene triangle has all its sides of different lengths.

The program must do the following:

1. read the 3 sides of the triangle from the keyboard; consider that the 3 sides are positive integer values and make up a valid triangle;
2. write on the screen the string "scalene", "isosceles" or "equilateral" depending on the length of the 3 sides.

Por exemplo:

Entrada	Resultado
5 5 5	equilateral
3 4 5	scalene
5 5 8	isosceles
12 10 10	isosceles
5 12 13	scalene

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)

```

1 #include <iostream>
2
3 using namespace std;
4
5 int main(){
6     int l1, l2, l3;
7     cin >> l1 >> l2 >> l3;
8     if(l1 == l2 && l2 == l3){
9         cout << "equilateral";
10    }
11    else if(l1 == l2 || l2 == l3 || l1 == l3){
12        cout << "isosceles";
13    }
14    else{
15        cout << "scalene";
16    }
17    return 0;
18 }
```

	Entrada	Esperado	Recebido	
✓	5 5 5	equilateral	equilateral	✓
✓	3 4 5	scalene	scalene	✓
✓	5 5 8	isosceles	isosceles	✓
✓	12 10 10	isosceles	isosceles	✓
✓	5 12 13	scalene	scalene	✓

Passou em todos os testes! ✓

Solução do autor da pergunta (Cpp):

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     //cout << "a b c ? ";
6     int a, b, c;
7     cin >> a >> b >> c;
8     if (a == b && a == c)
```

```
9      cout << "equilateral";
10  else if (a == b || a == c || b == c)
11      cout << "isosceles";
12  else
13      cout << "scalene";
14  return 0;
15 }
16
17 // private tests (1000 points each)
18 // 2 1 2 => isosceles
19 // 1000 1000 1000 => equilateral
20 // 20 15 7 => scalene
21 // 17 15 8 => scalene
22 // 255 255 256 => isosceles
```

Correta

Nota desta submissão: 20/20

Pergunta 2

Correta Pontuou 20 de 20

Write a C++ a function `void keep_prime_numbers(int a[], int& size)` that takes as parameters an array of positive integers, `a`, and its number of elements, `size`, and removes all the elements of the array that are not prime numbers, keeping the result in the same array. On exit, the parameter `size` must contain the effective number of elements.

Use the following function to determine whether a number is prime:

```
bool is_prime(int number) {
    if (number <= 1) return 0;
    for (int i = 2; i * i <= number; i++) {
        if (number % i == 0) return false;
    }
    return true;
}
```

Por exemplo:

Teste	Resultado
<pre>int a[] = { 1, 2, 3 }; int size = sizeof(a) / sizeof(int); keep_prime_numbers(a, size); print(a, size);</pre>	[2 3]
<pre>int a[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 }; int size = sizeof(a) / sizeof(int); keep_prime_numbers(a, size); print(a, size);</pre>	[2 3 5 7]
<pre>int a[] = { 2, 4, 6, 8, 10 }; int size = sizeof(a) / sizeof(int); keep_prime_numbers(a, size); print(a, size);</pre>	[2]
<pre>int a[] = { 5, 2, 9, 4, 14, 32, 64, 31 }; int size = sizeof(a) / sizeof(int); keep_prime_numbers(a, size); print(a, size);</pre>	[5 2 31]
<pre>int a[] = { 2, 2 }; int size = sizeof(a) / sizeof(int); keep_prime_numbers(a, size); print(a, size);</pre>	[2 2]

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)

Limpar resposta

```
1 #include <iostream>
2 using namespace std;
3
4 ///! Print array.
5 void print(int a[], int size) {
6     cout << "[ ";
7     for (int i = 0; i < size; i++)
8         cout << a[i] << " ";
9     cout << "]";
10    cout << endl;
11 }
12
13 ///! Determines whether 'number' is a prime number.
14 bool is_prime(int number) {
15     if (number <= 1) return 0;
16     for (int i = 2; i * i <= number; i++) {
17         if (number % i == 0) return false;
18     }
19     return true;
20 }
21
22 void keep_prime_numbers(int a[], int& size){
```

	Teste	Esperado	Recebido	
✓	int a[] = { 1, 2, 3 }; int size = sizeof(a) / sizeof(int); keep_prime_numbers(a, size); print(a, size);	[2 3]	[2 3]	✓
✓	int a[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 }; int size = sizeof(a) / sizeof(int); keep_prime_numbers(a, size); print(a, size);	[2 3 5 7]	[2 3 5 7]	✓
✓	int a[] = { 2, 4, 6, 8, 10 }; int size = sizeof(a) / sizeof(int); keep_prime_numbers(a, size); print(a, size);	[2]	[2]	✓
✓	int a[] = { 5, 2, 9, 4, 14, 32, 64, 31 }; int size = sizeof(a) / sizeof(int); keep_prime_numbers(a, size); print(a, size);	[5 2 31]	[5 2 31]	✓
✓	int a[] = { 2, 2 }; int size = sizeof(a) / sizeof(int); keep_prime_numbers(a, size); print(a, size);	[2 2]	[2 2]	✓

Passou em todos os testes! ✓

Solução do autor da pergunta (C):

```

1  #include <iostream>
2  using namespace std;
3
4  ///! Print array.
5  void print(int a[], int size) {
6      cout << "[ ";
7      for (int i = 0; i < size; i++)
8          cout << a[i] << " ";
9      cout << "]";
10     cout << endl;
11 }
12
13 ///! Determines whether 'number' is a prime number.
14 bool is_prime(int number) {
15     if (number <= 1) return 0;
16     for (int i = 2; i * i <= number; i++) {
17         if (number % i == 0) return false;
18     }
19     return true;
20 }
21
22 ///! Removes non-prime numbers from 'a' and updates 'size' accordingly.

```

Correta

Nota desta submissão: 20/20

Pergunta 3

Correta Pontuou 20 de 20

A character sequence in some alphabet is called a *heterogram* if each letter occurs at most once. For instance, "sun" is a heterogram and "moon" is not (the letter O occurs more than once in "moon").

Write a C++ function `bool heterogram(const char s[], char r[])` such that:

- `s` is a string containing uppercase or lowercase letter characters ('A' to 'Z', 'a' to 'z') and also the space character (' ') — spaces should be ignored and a lowercase character (e.g. 'a') should be considered equivalent to the corresponding uppercase letter (e.g., 'A');
- the function returns `true` if and only if the given string `s` is a heterogram; and
- on return, `r` is a lowercase string containing all letters that are repeated (occur more than once) in `s`, ordered alphabetically (`r` will be the empty string if `s` is a heterogram).

Hint: there are 26 letters in the alphabet. Use an internal array of length 26 to keep track of the letters that occur in `s`.

You cannot use any library classes or functions, including `vector`, `list`, `string`, `qsort` and `sort`.

Por exemplo:

Teste	Resultado
<pre>char s[] = ""; char r[27] = { -1 }; bool b = heterogram(s, r); cout << '\n' << s << "\n" << boolalpha << b << " \n" << r << "\n\n";</pre>	"" true ""
<pre>char s[] = "sun"; char r[27] = { -1 }; bool b = heterogram(s, r); cout << '\n' << s << "\n" << boolalpha << b << " \n" << r << "\n\n";</pre>	"sun" true ""
<pre>char s[] = "M0on"; char r[27] = { -1 }; bool b = heterogram(s, r); cout << '\n' << s << "\n" << boolalpha << b << " \n" << r << "\n\n";</pre>	"M0on" false "o"
<pre>char s[] = "C vs Python"; char r[27] = { -1 }; bool b = heterogram(s, r); cout << '\n' << s << "\n" << boolalpha << b << " \n" << r << "\n\n";</pre>	"C vs Python" true ""
<pre>char s[] = "Aha Zzz heterogram "; char r[27] = { -1 }; bool b = heterogram(s, r); cout << '\n' << s << "\n" << boolalpha << b << " \n" << r << "\n\n";</pre>	"Aha Zzz heterogram " false "aehrz"

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)

```
1 bool heterogram(const char s[], char r[]){
2     int alpha[26] = {0};
3     int i = 0;
4     while(s[i] != '\0'){
5         char c = s[i];
6         if('a' <= c && c <= 'z'){
7             int index = c - 'a';
8             alpha[index]++;
9         }
10        else if('A' <= c && c <= 'Z'){
11            int index = c - 'A';
12            alpha[index]++;
13        }
14        i++;
15    }
16    bool result = true;
17    int k = 0;
18    for(int i = 0; i < 26; i++){
19        if(alpha[i] > 1){
20            r[k++] = 'a' + i;
21            result = false;
22        }
23    }
```



	Teste	Esperado	Recebido	
✓	<pre>char s[] = ""; char r[27] = { -1 }; bool b = heterogram(s, r); cout << '\n' << s << "\n " << boolalpha << b << " \n" << r << "\n\n";</pre>	"" true ""	"" true ""	✓
✓	<pre>char s[] = "sun"; char r[27] = { -1 }; bool b = heterogram(s, r); cout << '\n' << s << "\n " << boolalpha << b << " \n" << r << "\n\n";</pre>	"sun" true ""	"sun" true ""	✓
✓	<pre>char s[] = "M0on"; char r[27] = { -1 }; bool b = heterogram(s, r); cout << '\n' << s << "\n " << boolalpha << b << " \n" << r << "\n\n";</pre>	"M0on" false "o"	"M0on" false "o"	✓
✓	<pre>char s[] = "C vs Python"; char r[27] = { -1 }; bool b = heterogram(s, r); cout << '\n' << s << "\n " << boolalpha << b << " \n" << r << "\n\n";</pre>	"C vs Python" true ""	"C vs Python" true ""	✓
✓	<pre>char s[] = " Aha Zzz heterogram "; char r[27] = { -1 }; bool b = heterogram(s, r); cout << '\n' << s << "\n " << boolalpha << b << " \n" << r << "\n\n";</pre>	" Aha Zzz heterogram " false "aehrz"	" Aha Zzz heterogram " false "aehrz"	✓

Passou em todos os testes! ✓

Solução do autor da pergunta (C):

```
1  //! Determines if a s-string is a heterogram.
2  bool heterogram(const char s[], char r[]) {
3      int count[26] = { 0 };
4      // add letters of s
5      for (int i = 0; s[i] != '\0'; i++) {
6          if (s[i] != ' ') {
7              if (s[i] >= 'a' && s[i] <= 'z')
8                  count[s[i] - 'a']++;
9              else
10                 count[s[i] - 'A']++;
11         }
12     }
13     // determines repeated letters
14     int n_r = 0;
15     for (char c = 'a'; c <= 'z'; c++) {
16         if (count[c - 'a'] > 1) {
17             r[n_r] = c;
18             n_r++;
19         }
20     }
21     r[n_r] = '\0';
22     return n_r == 0;
```

Correta

Nota desta submissão: 20/20

Pergunta 4

Correta Pontuou 20 de 20

Consider polynomials of the form

$$p(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_{n-1}x^{n-1}$$

where x and coefficients a_0, \dots, a_{n-1} are fractions described by type fraction as follows:

```
struct fraction {
    int num; // Numerator
    int den; // Denominator
};
```

Write a C++ function `fraction eval(const fraction a[], int n, fraction x)` that returns the fraction that results from evaluating for x the polynomial described by n coefficients stored in array a .

The result of the function should be an irreducible fraction and the denominator must always be positive. You may assume all fractions stored in a obey these conditions and that $n > 0$. A fraction n/d can be converted to irreducible form n'/d' by considering $n' = n/g$ and $d' = d/g$ where g is the greatest common divisor (g.c.d.) of n and d .

You can not use `pow` or other functions defined in `cmath` ou `math.h`.

You may use the following code for computing the g.c.d. of two numbers:

```
int gcd(int a, int b) {
    while (b != 0) {
        int tmp = a;
        a = b;
        b = tmp % b;
    }
    return a;
}
```

Hints:

- The problem should become simpler to solve if you start by defining auxiliary functions to implement the sum and the multiplication of fractions.
- Note that all elementary calculations involve integer numbers. Do not use floating point arithmetic; it is not required and it will not help you!

Por exemplo:

Teste	Resultado
<pre>const int n = 1; fraction p[n] = { { 1, 2 } }; fraction x = { 3, 4 }; fraction r = eval(p, n, x); cout << r.num << '/' << r.den << '\n';</pre>	1/2
<pre>const int n = 2; fraction p[n] = { { -3, 4 }, { -1, 2 } }; fraction x = { 0, 1 }; fraction r = eval(p, n, x); cout << r.num << '/' << r.den << '\n';</pre>	-3/4
<pre>const int n = 4; fraction p[n] = { { 1, 1 }, { 0, 1 }, { 0, 1 }, { 1, 1 } }; fraction x = { 2, 1 }; fraction r = eval(p, n, x); cout << r.num << '/' << r.den << '\n';</pre>	9/1
<pre>const int n = 3; fraction p[n] = { { 0, 1 }, { 1, 1 }, { 1, 2 } }; fraction x = { 1, 2 }; fraction r = eval(p, n, x); cout << r.num << '/' << r.den << '\n';</pre>	5/8
<pre>const int n = 4; fraction p[n] = { { -1, 2 }, { 1, 2 }, { -1, 2 }, { 1, 2 } }; fraction x = { -1, 3 }; fraction r = eval(p, n, x); cout << r.num << '/' << r.den << '\n';</pre>	-20/27

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)



[Limpar resposta](#)

```

1  #include <iostream>
2  using namespace std;
3
4  //! Fraction
5  struct fraction {
6      int num; // Numerator
7      int den; // Denominator
8  };
9
10  //! Compute the gcd of two numbers.
11  int gcd(int a, int b) {
12      while (b != 0) {
13          int tmp = a;
14          a = b;
15          b = tmp % b;
16      }
17      return a;
18  }
19  fraction mult(fraction f1, fraction f2){
20      int x = f1.num * f2.num;
21      int y = f1.den * f2.den;
22      int g = gcd(x,y);

```

	Teste	Esperado	Recebido	
✓	const int n = 1; fraction p[n] = { { 1, 2 } }; fraction x = { 3, 4 }; fraction r = eval(p, n, x); cout << r.num << '/' << r.den << '\n';	1/2	1/2	✓
✓	const int n = 2; fraction p[n] = { { -3, 4 }, { -1, 2 } }; fraction x = { 0, 1 }; fraction r = eval(p, n, x); cout << r.num << '/' << r.den << '\n';	-3/4	-3/4	✓
✓	const int n = 4; fraction p[n] = { { 1, 1 }, { 0, 1 }, { 0, 1 }, { 1, 1 } }; fraction x = { 2, 1 }; fraction r = eval(p, n, x); cout << r.num << '/' << r.den << '\n';	9/1	9/1	✓
✓	const int n = 3; fraction p[n] = { { 0, 1 }, { 1, 1 }, { 1, 2 } }; fraction x = { 1, 2 }; fraction r = eval(p, n, x); cout << r.num << '/' << r.den << '\n';	5/8	5/8	✓
✓	const int n = 4; fraction p[n] = { { -1, 2 }, { 1, 2 }, { -1, 2 }, { 1, 2 } }; fraction x = { -1, 3 }; fraction r = eval(p, n, x); cout << r.num << '/' << r.den << '\n';	-20/27	-20/27	✓

Passou em todos os testes! ✓

Solução do autor da pergunta (C):

```

1  #include <iostream>
2  using namespace std;
3
4  //! Fraction
5  struct fraction {
6      int num; // Numerator
7      int den; // Denominator
8  };
9
10  //! Compute the gcd of two numbers.
11  int gcd(int a, int b) {
12      while (b != 0) {
13          int tmp = a;
14          a = b;

```

```
15     b = tmp % b;  
16 }  
17 return a;  
18 }  
19 //! Normalize a fraction.  
20 fraction normalize(fraction f) {  
21     int g = gcd(f.num, f.den);  
22     int num = f.num / g;
```

Correta

Nota desta submissão: 20/20

Pergunta 5

Parcialmente correta Pontuou 4 de 20

Consider the code given in `node.cpp` containing the definition of type `node`, supporting the definition of doubly-linked lists with `int` values, and associated functions:

- `node* build(int v, node* n)`: builds a new node with value `v` (the `value` member), followed by `n` (the `next` member) — if `n != nullptr` then `n->prev` is set to point to the new node;
- `void destroy(node* n)`: releases the memory allocated to `n` and successor nodes; and
- `void print(const node* n)`: prints values in the node pointed by `n`.

Define a new function `node* remove_until(node* n, int v)` that removes all nodes from `n` until the first node that contains value `v`, and returns the resulting list. You must use `delete` appropriately to free memory. In the case where `v` does not occur in the list, all nodes should be removed.

You need to include `node.cpp` file in your code i.e. `#include "node.cpp"`.

You cannot use any C++ library classes or functions, including `vector`, `list`, or `string`.

Por exemplo:

Teste	Resultado
<pre>node* n = nullptr; int v = 0; n = remove_until(n, v); print(n); destroy(n);</pre>	
<pre>node* n = build(1, nullptr); int v = 1; n = remove_until(n, v); print(n); destroy(n);</pre>	(\<1<\)
<pre>node* n = build(2, build(1,nullptr)); int v = 1; n = remove_until(n, v); print(n); destroy(n);</pre>	(\<1<\)
<pre>node* n = build(5, build(4, build(3, build(3, build(2, build(1, nullptr)))))); int v = 3; n = remove_until(n, v); print(n); destroy(n);</pre>	(\<3<3)(3<3<2)(3<2<1)(2<1<\)
<pre>node* n = build(5, build(4, build(3, build(2, build(1, nullptr)))); int v = 6; n = remove_until(n, v); print(n); destroy(n);</pre>	

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)

Limpar resposta

```
1 | #include "node.cpp"
2 |
3 | node* remove_until(node* n, int v){
4 |     if(n == nullptr) return nullptr;
5 |     return n;
6 | }
```

↑

	Teste	Esperado	Recebido	
✓	node* n = nullptr; int v = 0; n = remove_until(n, v); print(n); destroy(n);			✓
✓	node* n = build(1, nullptr); int v = 1; n = remove_until(n, v); print(n); destroy(n);	(\<1<\)	(\<1<\)	✓
✗	node* n = build(2, build(1, nullptr)); int v = 1; n = remove_until(n, v); print(n); destroy(n);	(\<1<\)	(\<2<1) (2<1<\)	✗
✗	node* n = build(5, build(4, build(3, build(3, build(2, build(1, nullptr)))))); int v = 3; n = remove_until(n, v); print(n); destroy(n);	(\<3<3) (3<3<2) (3<2<1) (2<1<\)	(\<5<4) (5<4<3) (4<3<3) (3<3<2) (3<2<1) (2<1<\)	✗
✗	node* n = build(5, build(4, build(3, build(2, build(1, nullptr))))); int v = 6; n = remove_until(n, v); print(n); destroy(n);		(\<5<4) (5<4<3) (4<3<2) (3<2<1) (2<1<\)	✗

Alguns casos de teste escondidos também falharam.

Mostrar diferenças

Solução do autor da pergunta (C):

```

1 #include "node.cpp"
2
3 ///! Remove nodes until nod with the given value.
4 node* remove_until(node* n, int v) {
5     while (n != nullptr && n->value != v) {
6         node* tmp = n->next;
7         delete n;
8         n = tmp;
9     }
10    if (n != nullptr) {
11        n->prev = nullptr;
12    }
13    return n;
14 }
15
16 /*
17 // private tests (1000 points each)
18 {
19     node* n = build(1, nullptr);
20     int v = 99;
21     n = remove_until(n, v);
22     print(n);

```

Parcialmente correta

Nota desta submissão: 4/20

Ir para...

MT1: Revisão da teoria ►

