Início	terça, 19 de abril de 2022 às 14:34
Estado	Prova submetida
Data de submissão:	sábado, 23 de abril de 2022 às 11:24
Tempo gasto	3 dias 20 horas
Nota	**100** do máximo 100

Pergunta 1

Correta Pontuou 20 de 20

A character sequence in some alphabet is called a *pangram* if it contains every letter in the alphabet. For instance, "The quick brown fox jumps over the lazy dog" is a pangram in English since it contains all letters A to Z.

Write a C++ function `bool pangram(const string& s, string& m)` 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 pangram, that is, it contains all letters A to Z in lowercase or uppercase form; and
- on return, `m` is a lowercase string containing all letters that are missing in `s`, ordered alphabetically (`m` will be the empty string if `s` is a pangram).

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`.

Por exemplo:

Teste	Resultado

`string s = "";`
`string m = "";`
`bool r = pangram(s, m);`
`cout << '\"' << s << "\" "`
`  << boolalpha << r << " \"" << m << "\"\n";`
	`"" false "abcdefghijklmnopqrstuvwxyz"`

`string s = "The quick brown fox jumps over the lazy dog";`
`string m = "";`
`bool r = pangram(s, m);`
`cout << '\"' << s << "\" "`
`  << boolalpha << r << " \"" << m << "\"\n";`
	`"The quick brown fox jumps over the lazy dog" true ""`

`string s = "A quick brown fox jumps over a classy dog";`
`string m = "";`
`bool r = pangram(s, m);`
`cout << '\"' << s << "\" "`
`  << boolalpha << r << " \"" << m << "\"\n";`
	`"A quick brown fox jumps over a classy dog" false "htz"`

`string s = " abC dEf GhI jKl MnO pQr StU vWx yZ ";`
`string m = "";`
`bool r = pangram(s, m);`
`cout << '\"' << s << "\" "`
`  << boolalpha << r << " \"" << m << "\"\n";`
	`" abC dEf GhI jKl MnO pQr StU vWx yZ " true ""`

`string s = " Stu yZ abC GhI MnO pQr   ";`
`string m = "";`
`bool r = pangram(s, m);`
`cout << '\"' << s << "\" "`
`  << boolalpha << r << " \"" << m << "\"\n";`
	`" Stu yZ abC GhI MnO pQr   " false "defjklvwx"`

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

Limpar resposta

```
#include <string>
using namespace std;

void update_alpha(int index, int* alpha){
    alpha[index]++;
}

bool pangram(const string& s, string& m){

    int alpha[26] = {0};

    for(size_t i = 0; i < s.length(); i++){
        if ('a' <= s[i] && s[i] <= 'z'){
            update_alpha(s[i] - 'a', alpha);
        }
        if ('A' <= s[i] && s[i] <= 'Z'){
            update_alpha(s[i] - 'A', alpha);
        }
    }
    for(int i = 0; i < 26; i++){
```

```
        if(alpha[i] == 0)
            m += 'a' + i;
```

	Teste	Esperado	Recebido	

✔	`string s = "";`
	`string m = "";`
	`bool r = pangram(s, m);`
	`cout << '\"' << s << "\" "`
	`<< boolalpha << r << " \""`
	`<< m << "\"\n";`
		`"" false "abcdefghijklmnopqrstuvwxyz"`	`"" false "abcdefghijklmnopqrstuvwxyz"`	✔

✔	`string s = "The quick brown fox jumps over the lazy dog";`
	`string m = "";`
	`bool r = pangram(s, m);`
	`cout << '\"' << s << "\" "`
	`<< boolalpha << r << " \""`
	`<< m << "\"\n";`
		`"The quick brown fox jumps over the lazy dog" true ""`	`"The quick brown fox jumps over the lazy dog" true ""`	✔

✔	`string s = "A quick brown fox jumps over a classy dog";`
	`string m = "";`
	`bool r = pangram(s, m);`
	`cout << '\"' << s << "\" "`
	`<< boolalpha << r << " \""`
	`<< m << "\"\n";`
		`"A quick brown fox jumps over a classy dog" false "htz"`	`"A quick brown fox jumps over a classy dog" false "htz"`	✔

✔	`string s = " abC dEf GhI jKl MnO pQr StU vWx yZ ";`
	`string m = "";`
	`bool r = pangram(s, m);`
	`cout << '\"' << s << "\" "`
	`<< boolalpha << r << " \""`
	`<< m << "\"\n";`
		`" abC dEf GhI jKl MnO pQr StU vWx yZ " true ""`	`" abC dEf GhI jKl MnO pQr StU vWx yZ " true ""`	✔

✔	`string s = " Stu yZ abC GhI MnO pQr   ";`
	`string m = "";`
	`bool r = pangram(s, m);`
	`cout << '\"' << s << "\" "`
	`<< boolalpha << r << " \""`
	`<< m << "\"\n";`
		`" Stu yZ abC GhI MnO pQr   " false "defjklvwx"`	`" Stu yZ abC GhI MnO pQr   " false "defjklvwx"`	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
#include <string>
using namespace std;

//! Determines if a string is a pangram.
bool pangram(const std::string& s, string& missing_chars) {
  int count[26] = { 0 };
  for (size_t i = 0; i < s.length(); i++) {
    char c = s[i];
    if (c != ' ') {
      if (c >= 'a' && c <= 'z')
        count[c - 'a']++;
      else
        count[c - 'A']++;
    }
  }
  // determines missing letters
  bool r = true;
  for (char c = 'a'; c <= 'z'; c++) {
    if (count[c - 'a'] == 0) {
      missing_chars.push_back(c);
      r = false;
    }
```

Correta

Nota desta submissão: 20/20

Pergunta 2

Correta Pontuou 20 de 20

Write a C++ function `void split(const string& s, vector<string>& v)` that places in vector `v` all strings that occur in `s` separated by one or more spaces.

Hints: The class `string` member functions `find` and `substr` may be useful:

- `s.find(' ', pos)` searches for the space character in `s` starting from position `pos` — it returns the position of the first space found, or the constant `string::npos` if the space character is not found;
- `s.substr(pos, len)` produces the substring of `s` that starts at position `pos` and has `len` characters.

Por exemplo:

Teste	Resultado

`string s = "";`
`vector<string> v;`
`split(s, v);`
`print(v);`
	`[ ]`

`string s = "    ";`
`vector<string> v;`
`split(s, v);`
`print(v);`
	`[ ]`

`string s = "  a b  c ";`
`vector<string> v;`
`split(s, v);`
`print(v);`
	`[ "a" "b" "c" ]`

`string s = "C++ LEIC FCUP FEUP";`
`vector<string> v;`
`split(s, v);`
`print(v);`
	`[ "C++" "LEIC" "FCUP" "FEUP" ]`

`string s = "       C++        ";`
`vector<string> v;`
`split(s, v);`
`print(v);`
	`[ "C++" ]`

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

Limpar resposta

```
#include <string>
#include <vector>
#include <iostream>
using namespace std;

//! Print vector of strings.
void print(const vector<string>& v) {
  cout << "[ ";
  for (size_t i = 0; i < v.size(); i++) {
    cout << '\"' << v[i] << "\" ";
  }
  cout << "]\n";
}

void split(const string& s, vector<string>& v){
    size_t pos = s.find(" ", 0), last = 0;
    while(pos != string::npos){
        string new_str = s.substr(last, pos - last);
        if(new_str != "")
            v.push_back(new_str);
        last = pos+1;
        pos = s.find(" ",last);
```

	Teste	Esperado	Recebido	

	Teste	Esperado	Recebido	

✔	`string s = "";`
	`vector<string> v;`
	`split(s, v);`
	`print(v);`
		`[ ]`	`[ ]`	✔

✔	`string s = "    ";`
	`vector<string> v;`
	`split(s, v);`
	`print(v);`
		`[ ]`	`[ ]`	✔

✔	`string s = "  a b  c ";`
	`vector<string> v;`
	`split(s, v);`
	`print(v);`
		`[ "a" "b" "c" ]`	`[ "a" "b" "c" ]`	✔

✔	`string s = "C++ LEIC FCUP FEUP";`
	`vector<string> v;`
	`split(s, v);`
	`print(v);`
		`[ "C++" "LEIC" "FCUP" "FEUP" ]`	`[ "C++" "LEIC" "FCUP" "FEUP" ]`	✔

✔	`string s = "       C++        ";`
	`vector<string> v;`
	`split(s, v);`
	`print(v);`
		`[ "C++" ]`	`[ "C++" ]`	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
#include <string>
#include <vector>
#include <iostream>
using namespace std;

//! Print vector of strings.
void print(const vector<string>& v) {
  cout << "[ ";
  for (size_t i = 0; i < v.size(); i++) {
    cout << '\"' << v[i] << "\" ";
  }
  cout << "]\n";
}

//! Split string into vector of sub-strings.
void split(const string& s, vector<string>& v) {
  size_t pos = 0;
  while (pos < s.length()) {
    if (s[pos] == ' ') {
      pos++;
    } else {
      size_t end = s.find(' ', pos);
```

Correta

Nota desta submissão: 20/20

Pergunta 3

Correta Pontuou 20 de 20

Write a C++ template function

```
template <typename T>
void normalise(vector<T>& v, const T& min, const T& max)
```

such that a call to `normalise(v, min, max)` "normalises" all values `x` contained in vector `v` as follows:

- If `x < min` then `x` should be replaced by `min`;
- If `x > max` then `x` should be replaced by `max`; and
- Otherwise the value of `x` should be unchanged.

Por exemplo:

Teste	Resultado

`vector<int> v { };`
`normalise(v, 0, 1);`
`print(v);`
	`[ ]`

`vector<int> v { 1, 2, 3, 4, 5 };`
`normalise(v, 1, 5);`
`print(v);`
	`[ 1 2 3 4 5 ]`

`vector<int> v { 1, 2, 3, 4, 5 };`
`normalise(v, 3, 4);`
`print(v);`
	`[ 3 3 3 4 4 ]`

`vector<double> v { -1.2, 2.2, -3.5, 4.3, 5.2 };`
`normalise(v, 0.5, 5.1);`
`print(v);`
	`[ 0.5 2.2 0.5 4.3 5.1 ]`

`vector<string> v { "Diego", "Afonso", "Antonio", "Bernardo", "Tolentino", "Zeferino", "Xavier" };`
`normalise(v, string("Antonio"), string("Zacarias"));`
`print(v);`
	`[ Diego Antonio Antonio Bernardo Tolentino Zacarias Xavier ]`

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

Limpar resposta

```
#include <vector>
#include <iostream>
using namespace std;

//! Print vector with elements of type T.
template <typename T>
void print(const vector<T>& v) {
  cout << "[ ";
  for (size_t i = 0; i < v.size(); i++) {
    cout << v[i] << ' ';
  }
  cout << "]\n";
}

template <typename T>
void normalise(vector<T>& v, const T& min, const T& max){
    for(T& x : v){
        if(x < min) x = min;
        if(x > max) x = max;
    }
}
```

	Teste	Esperado	Recebido	

✔	`vector<int> v { };`
	`normalise(v, 0, 1);`
	`print(v);`
		`[ ]`	`[ ]`	✔

	Teste	Esperado	Recebido	

✔	`vector<int> v { 1, 2, 3, 4, 5 };`
	`normalise(v, 1, 5);`
	`print(v);`
		`[ 1 2 3 4 5 ]`	`[ 1 2 3 4 5 ]`	✔

✔	`vector<int> v { 1, 2, 3, 4, 5 };`
	`normalise(v, 3, 4);`
	`print(v);`
		`[ 3 3 3 4 4 ]`	`[ 3 3 3 4 4 ]`	✔

✔	`vector<double> v { -1.2, 2.2, -3.5, 4.3, 5.2 };`
	`normalise(v, 0.5, 5.1);`
	`print(v);`
		`[ 0.5 2.2 0.5 4.3 5.1 ]`	`[ 0.5 2.2 0.5 4.3 5.1 ]`	✔

✔	`vector<string> v { "Diego", "Afonso", "Antonio", "Bernardo", "Tolentino", "Zeferino", "Xavier" };`
	`normalise(v, string("Antonio"), string("Zacarias"));`
	`print(v);`
		`[ Diego Antonio Antonio Bernardo Tolentino Zacarias Xavier ]`	`[ Diego Antonio Antonio Bernardo Tolentino Zacarias Xavier ]`	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
#include <vector>
#include <iostream>
using namespace std;

//! Print vector with elements of type T.
template <typename T>
void print(const vector<T>& v) {
  cout << "[ ";
  for (size_t i = 0; i < v.size(); i++) {
    cout << v[i] << ' ';
  }
  cout << "]\n";
}

//! Normalise elements of vector for values between min and max.
template <typename T>
void normalise(vector<T>& v, const T& min, const T& max) {
  for (size_t i = 0; i < v.size(); i++) {
    T& e = v.at(i);  // or v[i]
    if (e > max)
      e = max;
    else if (e < min)
```

Correta

Nota desta submissão: 20/20

Pergunta 4

Correta Pontuou 20 de 20

Write a C++ function `string longest_prefix(const vector<string>& v)` that, given a vector `v` of strings, returns the longest common prefix to all of the strings.

For example, if `v` contains `"apple"`, `"apply"`, `"ape"`, and `"april"`, then the longest common prefix is `"ap"`.

Por exemplo:

Teste	Resultado
`cout << "\"" << longest_prefix({ "apple", "apply", "ape", "april" }) << "\"\n";`	`"ap"`
`cout << "\"" << longest_prefix({ "ap", "apple", "apply", "ape", "april", "aq", "x" }) << "\"\n";`	`""`
`cout << "\"" << longest_prefix({ "geezer", "geeksforgeeks", "geeks", "geek",  }) << "\"\n";`	`"gee"`
`cout << "\"" << longest_prefix({ "sedatesingratiateconcomitant", "sedatesparleypoliteness", "sedateselbowsHahn", "sedatesgloweringimbecility's", "sedatesbuttershexing", "sedatesKwangju'smulch's", "sedatesunwiserN", "sedatesprepossessedboggles", "sedatesinterrelationshipdialings", "sedatesgropesNelsen's", "sedatesMayfaircondemnations" }) << "\"\n";`	`"sedates"`
`cout << "\"" << longest_prefix({ "symmetricstalwartscorn'flowers", "symmetricShetland'ssuccessfully", "symmetricreceiver'shipsteed", "symmetricGrailsflown", "symmetricpromenadingoutflank", "symmetricdrugg'ingseizure", "symmetricsanctumsove'rprinting", "symmetricCitroentransgr'essed", "symmetricdissembledwinging", "symmetric'recoupingCinderella" }) << "\"\n";`	`"symmetric"`

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

Limpar resposta

```
#include <string>
#include <vector>
#include <iostream>
using namespace std;

string longest_prefix(const vector<string>& v){
    string res = v[0];
    for(string e : v){
        string aux = "";
        int i = 0;
        while(e[i] == res[i]){
            aux += e[i++];
        }
        res = aux;
    }
    return res;
}
```

	Teste	Esperado	Recebido	
✔	`cout << "\"" << longest_prefix({ "apple", "apply", "ape", "april" }) << "\"\n";`	`"ap"`	`"ap"`	✔
✔	`cout << "\"" << longest_prefix({ "ap", "apple", "apply", "ape", "april", "aq", "x" }) << "\"\n";`	`""`	`""`	✔
✔	`cout << "\"" << longest_prefix({ "geezer", "geeksforgeeks", "geeks", "geek", }) << "\"\n";`	`"gee"`	`"gee"`	✔
✔	`cout << "\"" << longest_prefix({ "sedatesingratiateconcomitant", "sedatesparleypoliteness", "sedateselbowsHahn", "sedatesgloweringimbecility's", "sedatesbuttershexing", "sedatesKwangju'smulch's", "sedatesunwiserN", "sedatesprepossessedboggles", "sedatesinterrelationshipdialings", "sedatesgropesNelsen's", "sedatesMayfaircondemnations" }) << "\"\n";`	`"sedates"`	`"sedates"`	✔

	Teste	Esperado	Recebido	
✔	`cout << "\"" << longest_prefix({ "symmetricstalwartscorn'flowers", "symmetricShetland'ssuccessfully", "symmetricreceiver'shipsteed", "symmetricGrailsflown", "symmetricpromenadingoutflank", "symmetricdrugg'ingseizure", "symmetricsanctumsove'rprinting", "symmetricCitroentransgr'essed", "symmetricdissembledwinging", "symmetric'recoupingCinderella" }) << "\"\n";`	`"symmetric"`	`"symmetric"`	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
#include <string>
#include <vector>
#include <iostream>
using namespace std;

//! Computes the longest common prefix to all of the strings.
string longest_prefix(const vector<string>& v) {
  if (v.size() == 0) {
    return "";
  }
  string prefix = v[0];
  for (size_t i = 1; i < v.size(); i++) {
    size_t pos = 0;
    while (pos < prefix.length() && pos < v[i].length()) {
      if (v[i][pos] != prefix[pos]) break;
      pos++;
    }
    if (pos == 0) {
      return "";
    }
    if (pos < prefix.length()) {
      prefix = prefix.substr(0, pos);
```

Correta

Nota desta submissão: 20/20

Pergunta 5

Correta Pontuou 20 de 20

A sparse matrix is a matrix where most elements have value 0. For space efficiency, data structures for sparse matrices only encode non-zero values.

Consider the following definition for a `smatrix` type, where sparse matrices are represented by a vector of type `sm_entry`:

```
struct sm_entry {
  size_t row;  // Matrix row
  size_t col;  // Matrix column
  int value;   // Value (non-zero)
};
typedef vector<sm_entry> smatrix;
```

Write a C++ function `void sum(const smatrix& a, const smatrix& b, smatrix& r)` such that a call to `sum(a, b, r)` places in `r` the result of adding the sparse matrices represented by `a` and `b`.

You should consider that entries in `a` and `b` are ordered by row first and then by column, and ensure the same type of ordering in `r`.

Por exemplo:

Teste	Resultado

`smatrix r;`
`sum({ },`
`  { {0, 3, 1}, {0, 50, 1} },`
`  r);`
`print(r);`
	`[ (0, 3, 1) (0, 50, 1) ]`

`smatrix r;`
`sum({ {0, 0, 1}, {1, 0, 1} },`
`  { {0, 3, 1}, {0, 50, 1} },`
`  r);`
`print(r);`
	`[ (0, 0, 1) (0, 3, 1) (0, 50, 1) (1, 0, 1) ]`

`smatrix r;`
`sum({ {0, 0, 1}, {0, 1, 2}, {5, 10, 20}, {99, 12, 32} },`
`  { {0, 0, 1}, {0, 1, -2}, {10, 5, 20}, {99, 10, 30}, {99, 11, 31} },`
`  r);`
`print(r);`
	`[ (0, 0, 2) (5, 10, 20) (10, 5, 20) (99, 10, 30) (99, 11, 31) (99, 12, 32) ]`

`smatrix r;`
`sum({ {0, 0, -1}, {0, 1, 2}, {5, 10, 20}, {10, 5, -20} },`
`  { {0, 0, 1}, {0, 1, -2}, {10, 5, 20} },`
`  r);`
`print(r);`
	`[ (5, 10, 20) ]`

`smatrix r;`
`sum({ {0, 0, -1}, {0, 1, 2},  {5, 10, 20 }, {10, 5, -20} },`
`  { {0, 0, 1}, {0, 1, -2},  {5, 10, -20}, {10, 5, 20} },`
`  r);`
`print(r);`
	`[ ]`

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

Limpar resposta

```
#include <string>
#include <vector>
#include <iostream>
using namespace std;

struct sm_entry {
  size_t row;
  size_t col;
  int value;
};

typedef vector<sm_entry> smatrix;

//! Print a sparse matrix
```

```
15 void print(const smatrix& sm) {
16   cout << "[ ";
17   for (sm_entry e : sm) {
18     cout << '(' << e.row << ", " << e.col << ", " << e.value << ") ";
19   }
20   cout << "]\n";
21 }
22
```

	Teste	Esperado	Recebido	

✔	`smatrix r;`
	`sum({ },`
	`  { {0, 3, 1}, {0, 50, 1} },`
	`  r);`
	`print(r);`
		`[ (0, 3, 1) (0, 50, 1) ]`	`[ (0, 3, 1) (0, 50, 1) ]`	✔

✔	`smatrix r;`
	`sum({ {0, 0, 1}, {1, 0, 1} },`
	`  { {0, 3, 1}, {0, 50, 1} },`
	`  r);`
	`print(r);`
		`[ (0, 0, 1) (0, 3, 1) (0, 50, 1) (1, 0, 1) ]`	`[ (0, 0, 1) (0, 3, 1) (0, 50, 1) (1, 0, 1) ]`	✔

✔	`smatrix r;`
	`sum({ {0, 0, 1}, {0, 1, 2}, {5, 10, 20}, {99, 12, 32} },`
	`  { {0, 0, 1}, {0, 1, -2}, {10, 5, 20}, {99, 10, 30}, {99, 11, 31} },`
	`  r);`
	`print(r);`
		`[ (0, 0, 2) (5, 10, 20) (10, 5, 20) (99, 10, 30) (99, 11, 31) (99, 12, 32) ]`	`[ (0, 0, 2) (5, 10, 20) (10, 5, 20) (99, 10, 30) (99, 11, 31) (99, 12, 32) ]`	✔

✔	`smatrix r;`
	`sum({ {0, 0, -1}, {0, 1, 2}, {5, 10, 20}, {10, 5, -20} },`
	`  { {0, 0, 1}, {0, 1, -2}, {10, 5, 20} },`
	`  r);`
	`print(r);`
		`[ (5, 10, 20) ]`	`[ (5, 10, 20) ]`	✔

✔	`smatrix r;`
	`sum({ {0, 0, -1}, {0, 1, 2}, {5, 10, 20 }, {10, 5, -20} },`
	`  { {0, 0, 1}, {0, 1, -2}, {5, 10, -20}, {10, 5, 20} },`
	`  r);`
	`print(r);`
		`[ ]`	`[ ]`	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
1  #include <string>
2  #include <vector>
3  #include <iostream>
4  using namespace std;
5
6  struct sm_entry {
7    size_t row;
8    size_t col;
9    int value;
10 };
11
12 typedef vector<sm_entry> smatrix;
13
14 void print(const smatrix& sm) {
15   cout << "[ ";
16   for (sm_entry e : sm) {
17     cout << '(' << e.row << ", " << e.col << ", " << e.value << ") ";
18   }
19   cout << "]\n";
20 }
21 // <-- Answer preload
22
```

Correta

Nota desta submissão: 20/20

◄ T05 05/04

Ir para...

T06 19/04 ►