Início	quarta, 23 de março de 2022 às 08:36
Estado	Prova submetida
Data de submissão:	terça, 29 de março de 2022 às 08:00
Tempo gasto	5 dias 22 horas
Nota	**100** do máximo 100

Pergunta 1

Correta Pontuou 20 de 20

Write a C++ function `unsigned long hstr_to_integer(const char hstr[])` that converts a C string formed by the characters that correspond to hexadecimal digits ('0' to '9', 'a' to 'f', or 'A' to 'F') to the corresponding integer decimal value. Remember that, as usual, C strings must be *null-terminated*.

You can not use any library functions or classes, including C++ classes `string`, `stringstream`, `vector`, `list` or C library functions `sscanf` and `strtol`. Your code can not have any `#include` directives.

Por exemplo:

Teste	Resultado
`cout << hstr_to_integer("0");`	0
`cout << hstr_to_integer("A");`	10
`cout << hstr_to_integer("19");`	25
`cout << hstr_to_integer("fF");`	255
`cout << hstr_to_integer("CafeBabe2022");`	223195403526178

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

```
unsigned long hstr_to_integer(const char hstr[]){
    unsigned long result = 0;
    int i = 0;
    while(hstr[i] != '\0'){
        result *= 16;
        if (hstr[i] >= '0' && hstr[i] <= '9'){
            result += (hstr[i] - 48);
        }
        else if(hstr[i] >= 'a' && hstr[i] <= 'f'){
            result += (hstr[i] - 97 + 10);
        }
        else if(hstr[i] >= 'A' && hstr[i] <= 'F'){
            result += (hstr[i] - 65 + 10);
        }
        i++;
    }
    return result;
}
```

	Teste	Esperado	Recebido	
✔	`cout << hstr_to_integer("0");`	0	0	✔
✔	`cout << hstr_to_integer("A");`	10	10	✔
✔	`cout << hstr_to_integer("19");`	25	25	✔
✔	`cout << hstr_to_integer("fF");`	255	255	✔
✔	`cout << hstr_to_integer("CafeBabe2022");`	223195403526178	223195403526178	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
//! Converts an hexadecimal number to the corresponding integer decimal value.
unsigned long hstr_to_integer(const char hstr[]) {
  unsigned long r = 0;
  int i = 0;
  while (hstr[i] != '\0') {
    char c = hstr[i];
    unsigned long v;
    if (c >= '0' && c <= '9') {
      v = c - '0';
    }
```

```
    else if (c >= 'A' && c <= 'F') {
      v = c + 10 - 'A';
    }
    else {
      v = c + 10 - 'a';
    }
    r = r * 16 + v;
    i++;
  }
  return r;
}

```

Correta

Nota desta submissão: 20/20

Pergunta 2

Correta Pontuou 20 de 20

Consider a type for integer fractions defined by

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

Write a C++ function `fraction sum(const fraction fa[], int n)` that returns the sum of all `n` fractions stored in array `fa`. The returned fraction should be irreducible and its denominator should always be positive; you may assume all fractions stored in `fa` obey these conditions and that `n > 0`.

Hint: The Euclidean algorithm for computing the greatest common divisor of two numbers can be helpful. It is expressed by the following recurrence:

$$\gcd(a,b) = \begin{cases} a & , \text{ if } b = 0 \\ \gcd(b, a\%b) & , \text{ if } b \neq 0 \end{cases}$$

Por exemplo:

Teste	Resultado

`fraction fa[] { {1, 2} };`
`fraction s = sum(fa, 1);`
`cout << s.num << '/' << s.den << "\n";`
	`1/2`

`fraction fa[] { {1, 2}, {-1, 3}};`
`fraction s = sum(fa, 2);`
`cout << s.num << '/' << s.den << "\n";`
	`1/6`

`fraction fa[] { {1, 2}, {-1, 3}, {-3, 4} };`
`fraction s = sum(fa, 3);`
`cout << s.num << '/' << s.den << "\n";`
	`-7/12`

`fraction fa[] { {-1, 4}, {1, 2}, {-1, 8}, {-1, 8} };`
`fraction s = sum(fa, 4);`
`cout << s.num << '/' << s.den << "\n";`
	`0/1`

`fraction fa[] { {0, 1}, {1, 2}, {-2, 3}, {3, 4}, {-4, 5}};`
`fraction s = sum(fa, 5);`
`cout << s.num << '/' << s.den << "\n";`
	`-13/60`

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

Limpar resposta

```
int gcd(int a, int b){
    return (b == 0) ? a : gcd(b, a%b);
}

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

fraction sum(const fraction fa[], int n){
    fraction result = {0, 1};
    for(int i = 0; i < n; i++){
        fraction f = fa[i];
        int x = result.num * f.den + result.den * f.num;
        int y = result.den * f.den;
        int g = gcd(x,y);
        x = x/g;
        y = y/g;
        result.num = (y>0) ? x: -x;
        result.den = (y>0) ? y: -y;
    }
```

	Teste	Esperado	Recebido	

	Teste	Esperado	Recebido	

✔	`fraction fa[] { {1, 2} };`
	`fraction s = sum(fa, 1);`
	`cout << s.num << '/' << s.den << "\n";`
		1/2	1/2	✔

✔	`fraction fa[] { {1, 2}, {-1, 3}};`
	`fraction s = sum(fa, 2);`
	`cout << s.num << '/' << s.den << "\n";`
		1/6	1/6	✔

✔	`fraction fa[] { {1, 2}, {-1, 3}, {-3, 4} };`
	`fraction s = sum(fa, 3);`
	`cout << s.num << '/' << s.den << "\n";`
		-7/12	-7/12	✔

✔	`fraction fa[] { {-1, 4}, {1, 2}, {-1, 8}, {-1, 8} };`
	`fraction s = sum(fa, 4);`
	`cout << s.num << '/' << s.den << "\n";`
		0/1	0/1	✔

✔	`fraction fa[] { {0, 1}, {1, 2}, {-2, 3}, {3, 4}, {-4, 5}};`
	`fraction s = sum(fa, 5);`
	`cout << s.num << '/' << s.den << "\n";`
		-13/60	-13/60	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```c
struct fraction {
  int num;
  int den;
};

//! Returns the Greatest Common Divisor (GCD) of two numbers.
int gcd_rec(int a, int b) {
  if (b == 0) return a;
  else return gcd_rec(b, a % b);
}

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

//! Returns the sum of all fractions as an irreducible fraction.
fraction sum(const fraction fa[], int n) {
```

Correta

Nota desta submissão: 20/20

Pergunta 3

Correta Pontuou 20 de 20

Write a C++ function `void merge_arrays(const int a[], int na, const int b[], int nb, int c[])` that merges 2 arrays, `a` with `na > 0` elements and `b`, with `nb > 0` elements, sorted in ascending order, placing the result in array `c`. The values in `c` must also be sorted in ascending order.

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

Por exemplo:

Teste	Resultado

`const int NA = 4, NB = 6;`
`int a[NA] = { 1, 2, 4, 7};`
`int b[NB] = { 0, 3, 5, 6, 8, 9};`
`int c[NA+NB];`
`merge_arrays(a, NA, b, NB, c);`
`print_array(c, NA+NB);`
	`[0,1,2,3,4,5,6,7,8,9]`

`const int NA = 6, NB = 4;`
`int a[NA] = { 0, 3, 5, 6, 8, 9};`
`int b[NB] = { 1, 2, 4, 7};`
`int c[NA+NB];`
`merge_arrays(a, NA, b, NB, c);`
`print_array(c, NA+NB);`
	`[0,1,2,3,4,5,6,7,8,9]`

`const int NA = 2, NB = 3;`
`int a[NA] = { 0, 1 };`
`int b[NB] = { 2, 3, 4 };`
`int c[NA+NB];`
`merge_arrays(a, NA, b, NB, c);`
`print_array(c, NA+NB);`
	`[0,1,2,3,4]`

`const int NA = 3, NB = 2;`
`int a[NA] = { 2, 3, 4 };`
`int b[NB] = { 0, 1 };`
`int c[NA+NB];`
`merge_arrays(a, NA, b, NB, c);`
`print_array(c, NA+NB);`
	`[0,1,2,3,4]`

`const int NA = 15, NB = 15;`
`int a[NA] = { 0, 1, 1, 2, 3, 4, 5, 6, 6, 6, 7, 8, 8, 9, 9 };`
`int b[NB] = { 0, 0, 1, 2, 3, 3, 4, 5, 5, 5, 6, 7, 7, 8, 9 };`
`int c[NA+NB];`
`merge_arrays(a, NA, b, NB, c);`
`print_array(c, NA+NB);`
	`[0,0,0,1,1,1,2,2,3,3,3,4,4,5,5,5,5,6,6,6,6,7,7,7,8,8,8,9,9,9]`

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

Limpar resposta

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

//! Auxiliary function to print n elements of an array.
void print_array(const int a[], int n){
  cout << '[' << a[0];
  for (int i = 1; i < n; i++) {
    cout << ',' << a[i];
  }
  cout << "]\n";
}

void merge_arrays(const int a[], int na, const int b[], int nb, int c[]){
    int i = 0, j = 0, k = 0;
    while(i < na && j < nb){
        if(a[i] < b[j]){
            c[k] = a[i];
            i++;
            k++;
        }
        else{
            c[k] = b[j];
```

	Teste	Esperado	Recebido

✔	`const int NA = 4, NB = 6;`
	`int a[NA] = { 1, 2, 4, 7};`
	`int b[NB] = { 0, 3, 5, 6, 8, 9};`
	`int c[NA+NB];`
	`merge_arrays(a, NA, b, NB, c);`
	`print_array(c, NA+NB);`
		`[0,1,2,3,4,5,6,7,8,9]`	`[0,1,2,3,4,5,6,7,8,9]`

✔	`const int NA = 6, NB = 4;`
	`int a[NA] = { 0, 3, 5, 6, 8, 9};`
	`int b[NB] = { 1, 2, 4, 7};`
	`int c[NA+NB];`
	`merge_arrays(a, NA, b, NB, c);`
	`print_array(c, NA+NB);`
		`[0,1,2,3,4,5,6,7,8,9]`	`[0,1,2,3,4,5,6,7,8,9]`

✔	`const int NA = 2, NB = 3;`
	`int a[NA] = { 0, 1 };`
	`int b[NB] = { 2, 3, 4 };`
	`int c[NA+NB];`
	`merge_arrays(a, NA, b, NB, c);`
	`print_array(c, NA+NB);`
		`[0,1,2,3,4]`	`[0,1,2,3,4]`

✔	`const int NA = 3, NB = 2;`
	`int a[NA] = { 2, 3, 4 };`
	`int b[NB] = { 0, 1 };`
	`int c[NA+NB];`
	`merge_arrays(a, NA, b, NB, c);`
	`print_array(c, NA+NB);`
		`[0,1,2,3,4]`	`[0,1,2,3,4]`

✔	`const int NA = 15, NB = 15;`
	`int a[NA] = { 0, 1, 1, 2, 3, 4, 5, 6, 6, 6, 7, 8, 8, 9, 9 };`
	`int b[NB] = { 0, 0, 1, 2, 3, 3, 4, 5, 5, 5, 6, 7, 7, 8, 9 };`
	`int c[NA+NB];`
	`merge_arrays(a, NA, b, NB, c);`
	`print_array(c, NA+NB);`
		`[0,0,0,1,1,1,2,2,3,3,3,4,4,5,5,5,5,6,6,6,6,7,7,7,8,8,8,9,9,9]`	`[0,0,0,1,1,1,2,2,3,3,3,4,4,5,5,`

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

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

//! Auxiliary function to print n elements of an array.
void print_array(const int a[], int n){
  cout << '[' << a[0];
  for (int i = 1; i < n; i++) {
    cout << ',' << a[i];
  }
  cout << "]\n";
}

//! Merge two s-orted arrays into a s-orted array.
void merge_arrays(const int a[], int na, const int b[], int nb, int c[])
{
  int i = 0, j = 0, k = 0;
  while (i < na && j < nb) {
    if (a[i] < b[j]) {
      c[k++] = a[i++];
    } else {
      c[k++] = b[j++];
    }
```

Correta

Nota desta submissão: 20/20

Pergunta 4

Correta Pontuou 20 de 20

Run-length encoding (RLE) is a form of data compression in which "runs" of data (sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count; this is most useful on data that contains many long runs. For example, the RLE encoding of “aaaaabbbbbbbbbbbcccd” is “5a11b3c1d”.

Write a C++ function `void rleEncode(const char str[], char rle[])` that takes the input C string `str` containing only non-digit characters (‘0’ to ‘9’ cannot occur) and computes the RLE encoding of `str` in `rle`.

You cannot use any library classes or functions, including the `string`, `stringstream`, `vector`, and `list` C++ classes or the `sprintf` function. Your code can not have any `#include` directives.

Hint: it can be useful to think first of a function `void int_to_string(int n, char str[], int& pos)` that writes the digits of `n` to `str` starting from position `pos`. On exit, `pos` contains the next usable position.

Por exemplo:

Teste	Resultado

`char rle[1] = { -1 };`
`rleEncode("", rle);`
`cout << rle << endl;`	

`char rle[2 + 1] = { -1, -1, -1 };`
`rleEncode("a", rle);`
`cout << rle << endl;`
	1a

`char rle[10 + 1] = { -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1};`
`rleEncode("abcde", rle);`
`cout << rle << endl;`
	1a1b1c1d1e

`char rle[9 + 1] = { -1, -1, -1, -1, -1, -1, -1, -1, -1, -1 };`
`rleEncode("aaaaabbbbbbbbbbbcccd", rle);`
`cout << rle << endl;`
	5a11b3c1d

`char rle[3 + 1] = { -1, -1, -1, -1 };`
`rleEncode("xxxxxxxxxxxxxxxxxxxx", rle);`
`cout << rle << endl;`
	20x

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

```
void int_to_str(int n, char str[], int& pos){
    int n2=1, i=0;
    int alg;

    //inverse number
    while(n>0){
        alg = n%10;
        n /= 10;
        n2 = n2 * 10 + alg;
        i++;
    }
    while (i>0){
        alg = n2 % 10;
        n2 /= 10;
        str[pos] = alg + '0';
        pos++;
        i--;
    }
    return;
}

void rleEncode(const char str[], char rle[]){
```

	Teste	Esperado	Recebido	

✔	`char rle[1] = { -1 };`
	`rleEncode("", rle);`
	`cout << rle << endl;`
				✔

✔	`char rle[2 + 1] = { -1, -1, -1 };`
	`rleEncode("a", rle);`
	`cout << rle << endl;`
		1a	1a	✔

	Teste	Esperado	Recebido	

✔	`char rle[10 + 1] = { -1, -1, -1, -1, -1, -1, -1, -1, -1, -1};`
	`rleEncode("abcde", rle);`
	`cout << rle << endl;`
		1a1b1c1d1e	1a1b1c1d1e	✔

✔	`char rle[9 + 1] = { -1, -1, -1, -1, -1, -1, -1, -1, -1 };`
	`rleEncode("aaaaabbbbbbbbbbcccd", rle);`
	`cout << rle << endl;`
		5a11b3c1d	5a11b3c1d	✔

✔	`char rle[3 + 1] = { -1, -1, -1, -1 };`
	`rleEncode("xxxxxxxxxxxxxxxxxxxx", rle);`
	`cout << rle << endl;`
		20x	20x	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
1  //! Encode integer into s-tring.
2  void int_to_str(int n, char str[], int& pos) {
3    int d = 0;  // digits used by decimal representation of n
4    for (int i = n; i != 0; i = i / 10){
5      d++;
6    }
7    for (int i = d, j = n; i > 0; i--) {
8      str[pos + i - 1] = '0' + j % 10;
9      j = j / 10;
10   }
11   pos = pos + d;
12 }
13
14 //! RLE encoding for n repetitions of char c.
15 void rleEncodeAux(int n, char c, char rle[], int& pos) {
16   if (n == 0) {
17     return;
18   }
19   int_to_str(n, rle, pos);
20   rle[pos] = c;
21   pos++;
22 }
```

Correta

Nota desta submissão: 20/20

Pergunta 5

Correta Pontuou 20 de 20

Consider types `time_of_day` and `interval` to represent the time of day with a precision of minutes, and time intervals defined by start and end times, as follows:

```
struct time_of_day {
  unsigned char h; // hours
  unsigned char m; // minutes
};
struct interval {
  time_of_day start; // start time
  time_of_day end;   // end time
};
```

Consider that an `interval` value `il` includes all times that are equal to or later than `il.start` and (strictly) earlier than `il.end`. For instance, if `il.start = { 12, 30}` (representing time 12:30) and `il.end = {14, 30}` (14:30), then `{12, 30}` and `{12, 31}` are part of the interval but `{14, 30}` and `{14, 31}` are not.

Write a C++ function

`int search_intervals(time_of_day t, const interval a[], int n, interval& u)`

that searches for all `n` intervals stored in array `a`, and:

- If there are no intervals that contain `t`, then the function assigns u with `{ t, t }` and returns `0`.
- If there are intervals that contain `t`, it assigns `u` to the union of all these intervals and returns the total duration of `u` in minutes.

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

Por exemplo:

Teste	Resultado

`const int n = 1;`
`const time_of_day t = { 13, 00 };`
`interval a[n] { { { 12, 30 }, { 14, 30 } } };`
`interval u;`
`int d = search_intervals(t, a, n, u);`
`print(d, u);`
	`120 [12:30,14:30[`

`const int n = 2;`
`const time_of_day t = { 14, 30 };`
`interval a[n] { { { 12, 30 }, { 14, 30 } },`
`{ { 14, 30 }, { 15, 30 } } };`
`interval u;`
`int d = search_intervals(t, a, n, u);`
`print(d, u);`
	`60 [14:30,15:30[`

`const int n = 2;`
`const time_of_day t = { 12, 30 };`
`interval a[n] { { { 12, 30 }, { 14, 30 } },`
`{ { 14, 30 }, { 15, 30 } } };`
`interval u;`
`int d = search_intervals(t, a, n, u);`
`print(d, u);`
	`120 [12:30,14:30[`

`const int n = 2;`
`const time_of_day t = { 15, 30 };`
`interval a[n] { { { 12, 30 }, { 14, 30 } },`
`{ { 14, 30 }, { 15, 30 } } };`
`interval u;`
`int d = search_intervals(t, a, n, u);`
`print(d, u);`
	`0 [15:30,15:30[`

`const int n = 5;`
`const time_of_day t = { 15, 15 };`
`interval a[n] { { { 12, 30 }, { 14, 30 } },`
`{ { 14, 30 }, { 15, 30 } },`
`{ { 15, 10 }, { 16, 10 } },`
`{ {  9, 30 }, { 15, 15 } },`
`{ {  9, 45 }, { 15, 16 } } };`
`interval u;`
`int d = search_intervals(t, a, n, u);`
`print(d, u);`
	`385 [9:45,16:10[`

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

Limpar resposta

```
struct time_of_day {
  unsigned char h;  // hours (0 to 23)
  unsigned char m;  // minutes (0 to 59)
};
struct interval {
  time_of_day start;  // start time
  time_of_day end;    // end time
};
#include <iostream>
using namespace std;

//! Prints the results.
void print(int d, const interval il) {
  cout << d << " ["
       << (int) il.start.h << ':' << (int) il.start.m << ','
       << (int) il.end.h   << ':' << (int) il.end.m
       << "[\n";
}
int search_intervals(time_of_day t, const interval a[], int n, interval& u){
    time_of_day min_u = {23, 59}, max_u = {0, 0};
    bool found = false;
    for(int i = 0; i < n; i++){
```

	Teste	Esperado	Recebido	

✔	`const int n = 1;`
	`const time_of_day t = { 13, 00 };`
	`interval a[n] { { { 12, 30 }, { 14, 30 } } };`
	`interval u;`
	`int d = search_intervals(t, a, n, u);`
	`print(d, u);`
		`120 [12:30,14:30[`	`120 [12:30,14:30[`	✔

✔	`const int n = 2;`
	`const time_of_day t = { 14, 30 };`
	`interval a[n] { { { 12, 30 }, { 14, 30 } },`
	`{ { 14, 30 }, { 15, 30 } } };`
	`interval u;`
	`int d = search_intervals(t, a, n, u);`
	`print(d, u);`
		`60 [14:30,15:30[`	`60 [14:30,15:30[`	✔

✔	`const int n = 2;`
	`const time_of_day t = { 12, 30 };`
	`interval a[n] { { { 12, 30 }, { 14, 30 } },`
	`{ { 14, 30 }, { 15, 30 } } };`
	`interval u;`
	`int d = search_intervals(t, a, n, u);`
	`print(d, u);`
		`120 [12:30,14:30[`	`120 [12:30,14:30[`	✔

✔	`const int n = 2;`
	`const time_of_day t = { 15, 30 };`
	`interval a[n] { { { 12, 30 }, { 14, 30 } },`
	`{ { 14, 30 }, { 15, 30 } } };`
	`interval u;`
	`int d = search_intervals(t, a, n, u);`
	`print(d, u);`
		`0 [15:30,15:30[`	`0 [15:30,15:30[`	✔

✔	`const int n = 5;`
	`const time_of_day t = { 15, 15 };`
	`interval a[n] { { { 12, 30 }, { 14, 30 } },`
	`{ { 14, 30 }, { 15, 30 } },`
	`{ { 15, 10 }, { 16, 10 } },`
	`{ {  9, 30 }, { 15, 15 } },`
	`{ {  9, 45 }, { 15, 16 } } };`
	`interval u;`
	`int d = search_intervals(t, a, n, u);`
	`print(d, u);`
		`385 [9:45,16:10[`	`385 [9:45,16:10[`	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
struct time_of_day {
  unsigned char h;  // hours (0 to 23)
  unsigned char m;  // minutes (0 to 59)
};
struct interval {
  time_of_day start;  // start time
  time_of_day end;    // end time
};
#include <iostream>
using namespace std;

//! Prints the results.
void print(int d, const interval il) {
  cout << d << " ["
       << (int) il.start.h << ':' << (int) il.start.m << ','
       << (int) il.end.h   << ':' << (int) il.end.m
       << "[\n";
}
//! Compares two times of day.
bool earlier_or_equal(time_of_day t1, time_of_day t2) {
  return t1.h < t2.h || (t1.h == t2.h && t1.m <= t2.m);
}
```

Correta

Nota desta submissão: 20/20

◄ T02 15/03

Ir para...

T03 22/03 ►