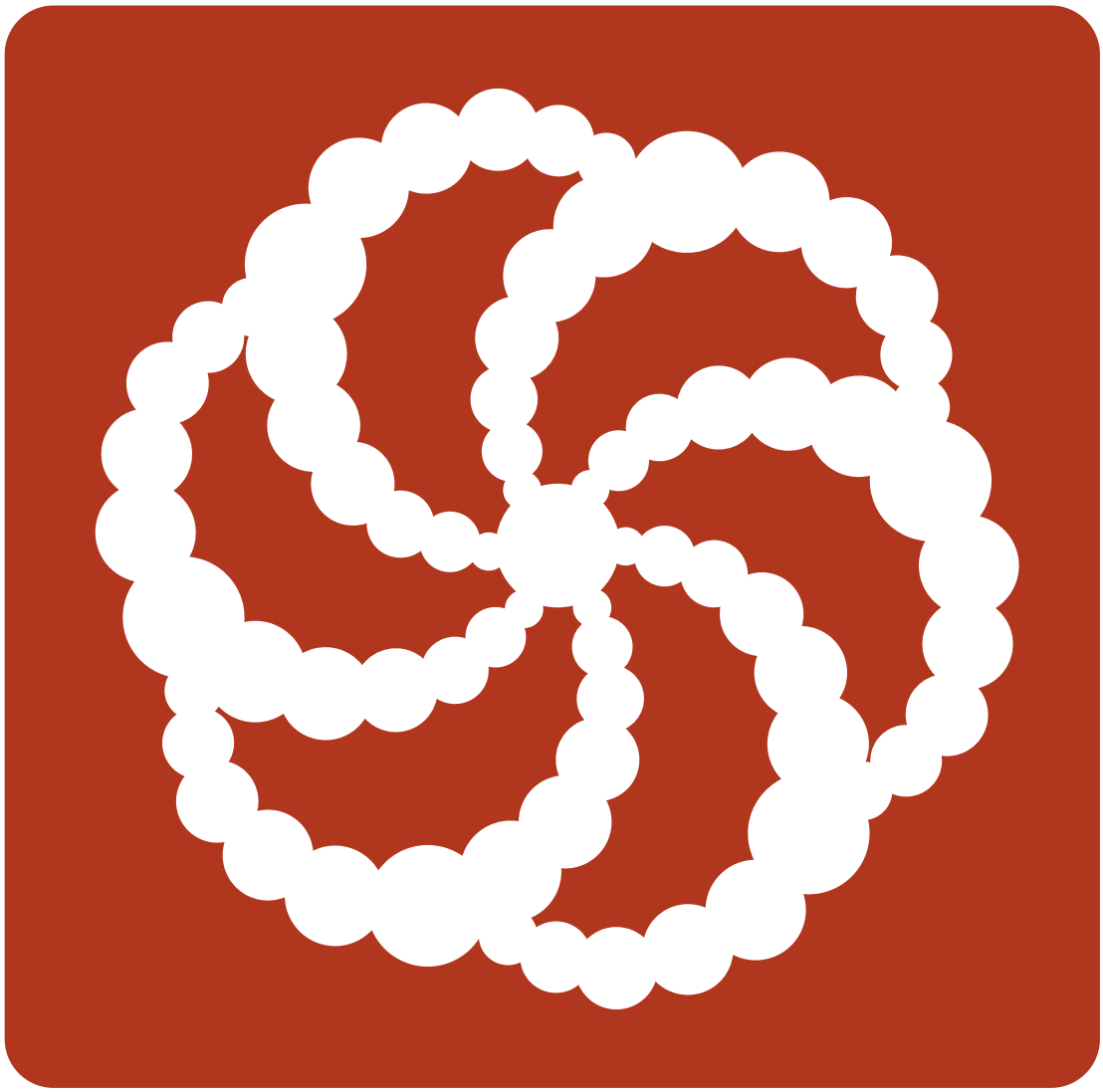
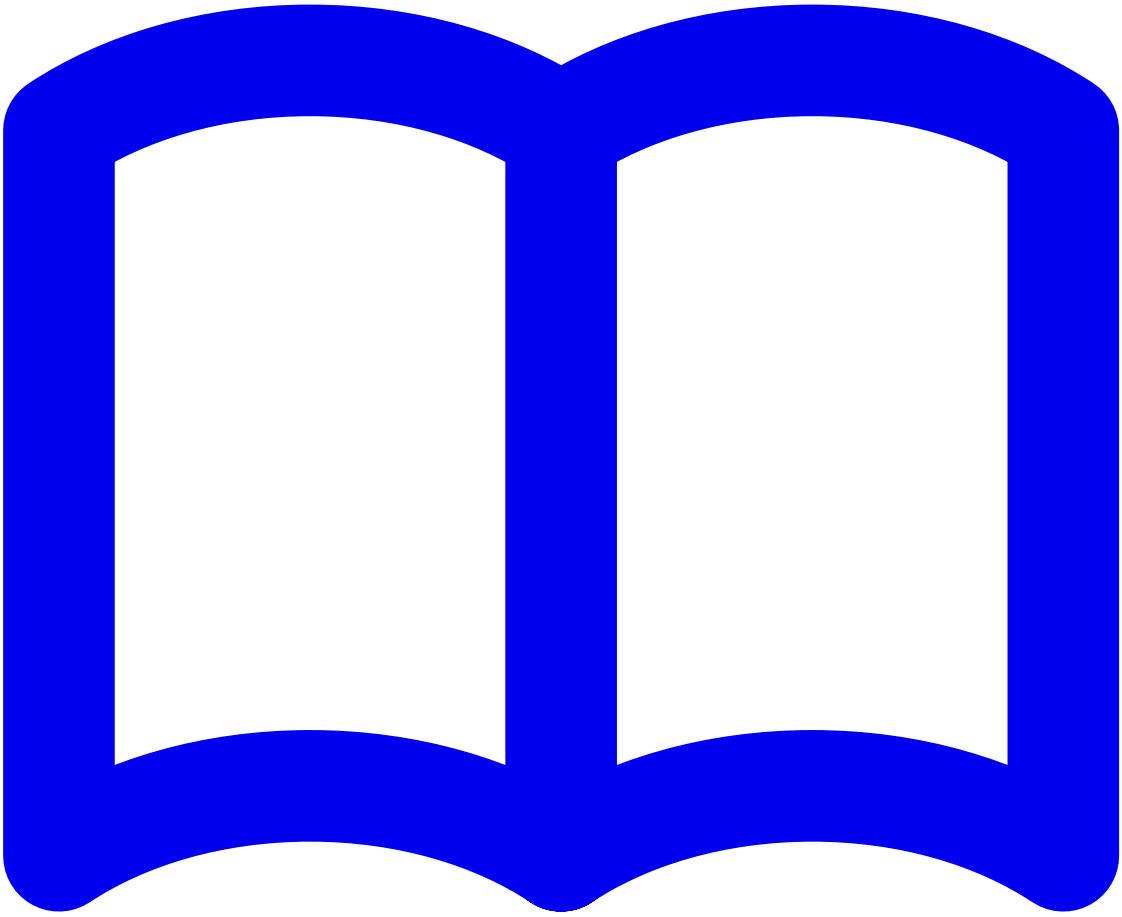
Kata

- 

  Painel
-

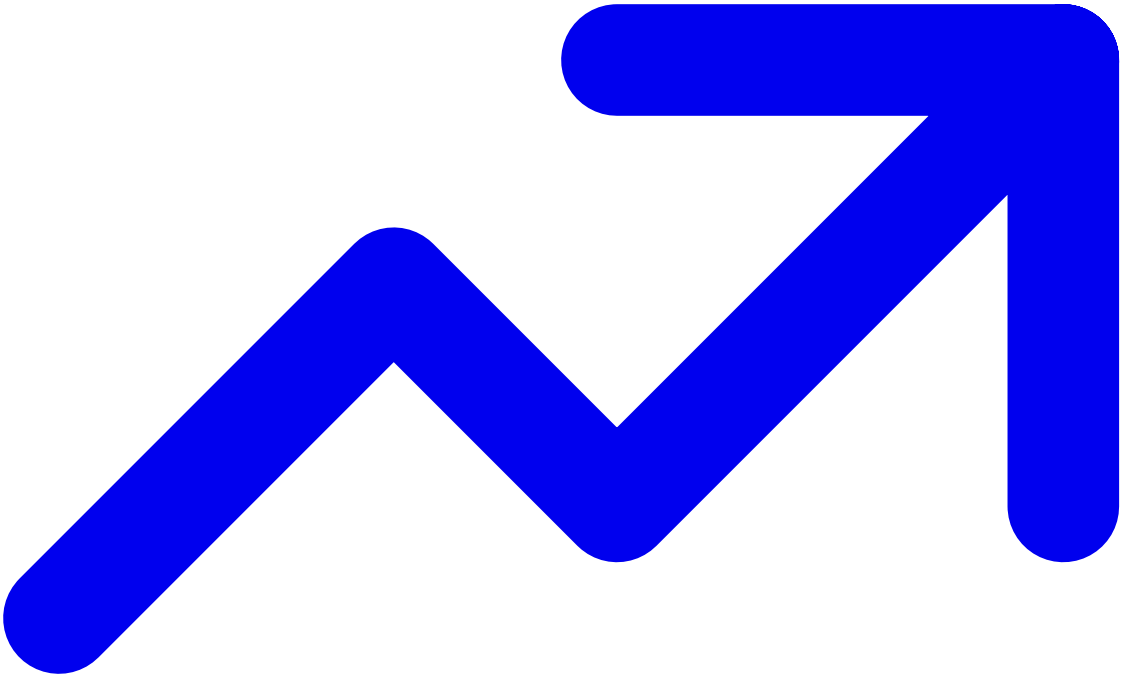
- Kata



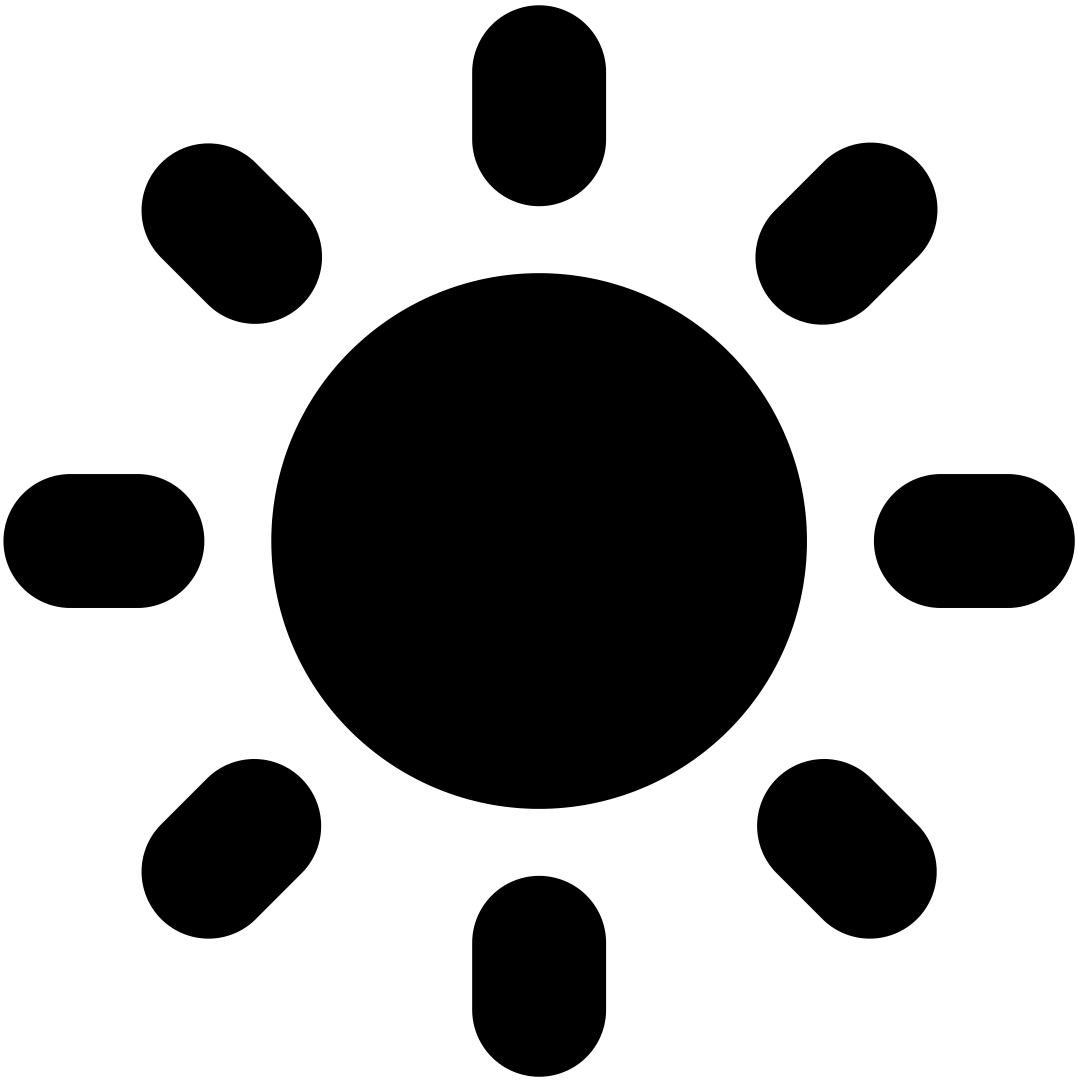
- Docs

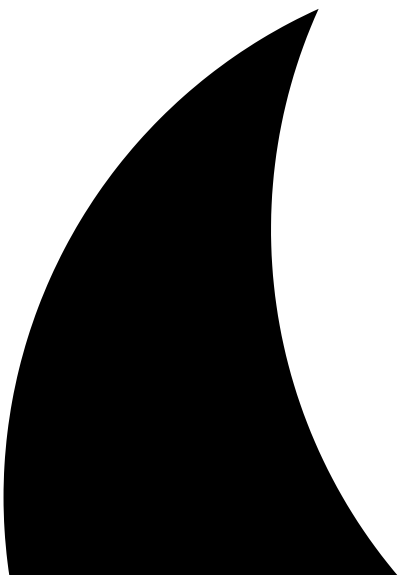
- Blog
- Kumite
-

Fórum



Líderes





- Você não marcou nenhum kata com estrela
  Para adicionar alguns, basta clicar nopróximo a qualquer título de kata.
- Você não tem nenhuma notificação
- 
  8 kyu
  8
  - Ver perfil
  - Configurações da conta
  - Configuração de treinamento
  - Atualize para vermelho
  - Sair

7 kyu

Soma do primeiro enésimo termo da Série

86186113685% de 5.5621.161 de 46.141DivyanshBatham
C
Escolha o seu idioma...

C
Clojure
CoffeeScript
C ++
C #
Elixir
Haskell
Java
JavaScript
PHP
Pitão
Rubi
Scala
TypeScript
Adicionar novo

Treinar de novoPróximo Kata

- Detalhes
- Soluções
- Garfos (14)
- Discurso (395)
- Traduções

Colete |
Mostrar descrição do Kata

Descrição:

Carregando descrição ...
Fundamentos
rotações
Controle de fluxo
Recursos básicos da linguagem
Arithmetic
Mathematics
Algorithms
Numbers
Sequences
Arrays

Suggest kata description edits

Mostrar Casos de Teste Kata

Test Cases:

```
#include <criterion/criterion.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

char *series_sum(const size_t n);

char *series_sum_tester(const size_t n);

size_t getRandomSizeTypeInteger(size_t lowerBoundInclusive, size_t upperBoundExclusive);

Test(BasicTests, ShouldPassAllTheTestsProvided) {
  {
    const char *expected = "1.00";
    char *received = series_sum(1);
    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);
    free(received); received = NULL;
  }

  {
    const char *expected = "1.25";
    char *received = series_sum(2);
    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);
    free(received); received = NULL;
  }

  {
    const char *expected = "1.39";
    char *received = series_sum(3);
    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);
    free(received); received = NULL;
  }

  {
    const char *expected = "1.49";
    char *received = series_sum(4);
    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);
    free(received); received = NULL;
  }

  {
    const char *expected = "1.57";
    char *received = series_sum(5);
    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);
    free(received); received = NULL;
  }

  {
    const char *expected = "1.63";
    char *received = series_sum(6);
    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);
    free(received); received = NULL;
  }
```

```
  {
    const char *expected = "1.00";
    char *received = series_sum(1);
    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);
    free(received); received = NULL;
  }

  {
    const char *expected = "1.68";
    char *received = series_sum(7);
    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);
    free(received); received = NULL;
  }

  {
    const char *expected = "1.73";
    char *received = series_sum(8);
    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);
    free(received); received = NULL;
  }

  {
    const char *expected = "1.77";
    char *received = series_sum(9);
    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);
    free(received); received = NULL;
  }

  {
    const char *expected = "1.94";
    char *received = series_sum(15);
    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);
    free(received); received = NULL;
  }

  {
    const char *expected = "2.26";
    char *received = series_sum(39);
    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);
    free(received); received = NULL;
  }

  {
    const char *expected = "2.40";
    char *received = series_sum(58);
    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);
    free(received); received = NULL;
  }

  {
    const char *expected = "0.00";
    char *received = series_sum(0);
    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);
    free(received); received = NULL;
  }
}

Test(RandomTests, ShouldPassAllTheTestsProvided) {
  const size_t TEST_COUNT = 40;
  const size_t MIN_N_VALUE = 0;
  const size_t MAX_N_VALUE = 100;

  srand((unsigned)time(NULL));

  for (size_t times = 0; times < TEST_COUNT; ++times) {
    const size_t n = getRandomSizeTypeInteger(MIN_N_VALUE, MAX_N_VALUE + 1);

    char *expected = series_sum_tester(n);
    char *received = series_sum(n);

    cr_assert_str_eq(expected, received, "Expected: %s Received: %s", expected, received);

    free(expected); expected = NULL;
    free(received); received = NULL;
  }
}

size_t getRandomSizeTypeInteger(size_t lowerBoundInclusive, size_t upperBoundExclusive) {
  return (size_t)rand() % (upperBoundExclusive - lowerBoundInclusive) + lowerBoundInclusive;
}
```

```
char *series_sum_tester(const size_t n) {
  const size_t count = 32;
  char *buffer = (char*)malloc(sizeof(char) * count);
  if (buffer) {
    double value = 0.0;
    int divisor = 1;
    int step = 3;
    for (size_t times = 0; times < n; ++times) {
      value += 1.0 / divisor;
      divisor += step;
    }

    (void)snprintf(buffer, count, "%.2lf", value);
  }
  return buffer;
}
```

Suggest test case edits

- **Mostre-me:**
- Todas as Soluções
- Soluções de usuários que estou seguindo

- **Ordenar por:**
- Melhores Práticas
- Inteligente
- O mais novo
- Mais velho

- **teleparioSharan NDavidSoler**

```
char *series_sum(const int n) {
  double result = n ? 1. : 0.;
  char *s = (char *)malloc(10 * sizeof(char));

  for (int i = 1; i < n; i++) result += 1. / (3 * i + 1);
  sprintf(s, "%.2lf", result);
  return s;
}
```

2 variações de código semelhantes são agrupadas com esteMostrar Variações

- - Melhores Práticas32
  - Inteligente12
- 4
- 1
- Garfo
- Link
- 

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content
- Post
- Cancel

- crees(8 kyu)
- 3 months ago

sprintf can't cause buffer overflow with this series, even with malloc(5*sizeof(char)) as the sum to infinity is below 10.

- 1|
- Reply
- View Solution
- Spoiler

- neuroQuanta(6 kyu)
- 2 years ago

sprintf may cause a buffer overflow (with this sum series not likely tho) => better use snprintf(..)

- -1|
- Reply
- View Solution
- Spoiler

- jokerkeny(7 kyu)
- 2 years ago

Actually there is no need to use n ? 1.: 0.;
you can start for as int i = 0

- 4|
- Reply
- View Solution
- Spoiler

- drozdziak1(8 kyu)
- 4 years ago

Casting malloc() in C is always a bad idea. `void *` allows for implicit casts to any type, effectively making you only repeat yourself. What is more, if you were to also forget `#include <stdlib.h>` in real-life code, your compiler could take `malloc` as an implicit function, which by default is presumed to return an `int`. That could go very well undetected on many setups, but if your system's ints have a smaller size than pointers, you might end up with malloc()'s spewing ill-formed addresses.

sizeof(char) is basically useless, as char is guaranteed to always be 1 byte wide.

Are you sure you need 10 characters? I wouldn't make the size constant, but the output is *always* something like "1.xx" and it's never reaching even "2.xx". How about going with snprintf() instead?

Checking if `n == 0` is not necessary, your for loop can take care of that. Initialize `result` to `.0` right away and start the `for` loop at `int i = 0`, the `n = 1` case will solve itself with a single `for` iteration resolving `1. / (3 * i + 1)` to `1/1`.

- 6|
- Reply
- View Solution
- Spoiler

- **Sem nome**

```
#define _GNU_SOURCE
#include <stddef.h>
#include <stdio.h>

char *series_sum(const size_t n) {
  double sum = 0;
  for (double d = 1 + 3 * ((double)n - 1); d >= 1; d -= 3)
    sum += 1 / d;
  char *res = NULL;
  asprintf(&res, "%.2f", sum);
  return res;
}
```

  - Melhores Práticas8
  - Inteligente4
  - 4
  - Garfo
  - Link
  - 

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content

- Post
- Cancel

  - 

- **swerrock**(7 kyu)
- last year

asprintf usage was perfect fit.

- 1|
- Reply
- View Solution
- Spoiler

- GithubIsAwful(5 kyu)
- 2 years ago

isn't `char *res` just writing into an arbitrary position in memory?

- 1|
- Reply
- View Solution
- Collapse
- Spoiler

- Unnamed(1 dan)
- 2 years ago

It's not the standard C `sprintf`, it's a GNU extension for cases like this: https://linux.die.net/man/3/asprintf

- 1|
- View Solution
- Spoiler

- GithubIsAwful(5 kyu)
- 2 years ago

Didn't see that, thanks.

- 1|
- View Solution
- Spoiler

- Reply

- WestonP22

```
#include <stddef.h>

char *series_sum(const size_t n)
{
  size_t i;
  int divisor;
  double sum;
  char * result;

  for(i = 0, divisor = 1, sum = 0.0; i < n; ++i, divisor += 3)
  {
    sum += 1.0 / (double)divisor;
  }

  result = malloc(32);

  if(!result)
    return NULL;

  sprintf(result, "%.02f", sum);
```

```
  return result;
}
```

- Melhores Práticas6
- Inteligente3
- 0
- Garfo
- Link
- 

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content
- Post
- Cancel

- **Kamillamagna**

```
#include <stddef.h>

char *series_sum(const size_t n) {            // n = size of series
  float sum = 0.0000;                         // initialize sum to float 0 for float addition
  char* sumStr = calloc(16, sizeof(char));    // allocate 16 0-bytes for result string
  float denom = 1.0000;                       // initialize denominator to float 1 for float division
  for (int i = 0; i < n; i++) {               // n times do:
    sum += ( 1.0000 / denom );                //   add series term i to sum
    denom += (float) 3;                       //   set denominator for next iteration
  }
  sprintf(                                    // format string:
    sumStr,                                   //   at memory location sumStr
    "%.2f",                                   //   float f in decimal format with 2 places
    sum                                       //   where f is sum calculated above
  );
  return sumStr;                              // sumStr
}
```

- Melhores Práticas5
- Inteligente0
- 0
- Garfo
- Link
- 

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content
- Post
- Cancel

- jaqob

```
#include <stddef.h>

char *series_sum(const size_t n)
{
  double sum = 0;
  double base = 1;
  for(int i = 0; i < n; i++)
  {
    sum+=(1/base);
    base+=3;
  }

  sum = roundf(sum * 100) / 100;

  char *output = malloc(5);

snprintf(output, 5, "%f", sum);

  return output;
}
```

- Melhores Práticas3
- Inteligente0
- 0
- Garfo
- Link

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content

- Post
- Cancel

- cg18

```
#include <stddef.h>
#include<string.h>
#include<stdlib.h>
char *series_sum(const size_t n)
{double sum=0;
 char*p;p=(char*)malloc(4);
 if(n==0)
 {*p=*(p+2)=*(p+3)='0'; *(p+1)='.'; return p;}
 for(int i=1;i<=n;i++)
  sum+=1.0/(3*i-2);
  *p='0'+(int)sum;
  *(p+1)='.';
  sum*=10; int m=(int)sum%10;
  sum*=10; int k=(int)sum%10;
  sum*=10; int j=(int)sum%10;
  if(j>=5) k++;
  if(k==10) {k=0;m++;}
  *(p+2)='0'+m;
  *(p+3)='0'+k;
  return p;
  }
```

- Melhores Práticas2
- Inteligente3

- 1
- Garfo
- Link
- 

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content
- Post
- Cancel

- 

- **dialbuzzer(6 kyu)**
- **4 years ago**

Looks like you're trying to apply for "The International Obfuscated C Code Contest" (look it up if you don't know what it is).
No offense :)

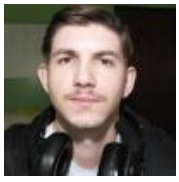
- 6|
- Reply
- View Solution
- Spoiler

- **telepário**

```
char *series_sum(const int n) {
  double result = n ? 1. : 0.;
  char *s = (char *)malloc(50 * sizeof(char));

  for (int i = 1; i < n; i++) result += 1. / (3 * i + 1);
  sprintf(s, "%.2lf", result);
  return s;
}
```

- Melhores Práticas2
- Inteligente0

- 0
- Garfo
- Link
- 

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content
- Post
- Cancel

- **Wolfie_**

```
#include <stddef.h>
#include <stdlib.h>
#include <stdio.h>

char *series_sum(const size_t n){
  char *str = (char *)malloc(sizeof(float));
  float sum = (int)n==0?0:1 ;
  for (float i = 4 , m = 1 ; m < (float)n ; m ++ , i+=3 ){
    sum += (float)(1/i);
  }
  sprintf(str,"%.2f",sum);
  return str;
}
```

- Melhores Práticas1
- Inteligente0
- 0
- Garfo
- Link
- 

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content
- Post
- Cancel

- **99camille**

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>

char *series_sum(const size_t n)
{
    float sum = 0.00;
    for (size_t i = 0; i < n; i++)
        sum += 1.00 / (1 + i * 3);

    char *ret = (char*) malloc(5);
    sprintf(ret, "%.2f", sum);

    return ret;
}
```

- Melhores Práticas1
- Inteligente0

- 0
- Garfo
- Link
- 

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content
- Post
- Cancel

- **Gena2018115rus**

```
#include <stddef.h>

char *series_sum(const size_t n)
{
    double sum = 0.0;
    for (size_t i = 0; i < n; ++i) sum += 1.0 / (i * 3 + 1);
    asprintf(&sum, "%.2lf", sum);
    return *(void **)&sum;
}
```

- Melhores Práticas1
- Inteligente0
- 0
- Garfo
- Link
- 

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content
- Post
- Cancel

- **testvanilla0118**

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char *series_sum(const size_t n) {
  double sum = 0.00;
```

```
  char *buffer = malloc(sizeof(char) * (50));
  for(int i = 0; i < n; i++) {
    sum += (1.00 / (1.00 + 3 * i));
  }

  snprintf(buffer, 50, "%.2f", sum);
  return buffer;
}
```

- Best Practices1
- Clever0
- 0
- Fork
- Link
- 

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content

- Post
- Cancel

- **halforc**

```
#include <stddef.h>

char *series_sum(const size_t n) {
  double s = 0;
  char *ret = malloc(5);
  for (size_t i = 0; i < n; i++)
    s += 1.0 / (1 + i * 3);
  snprintf(ret, 5, "%1.2f", s);
  return ret;
}
```

- Best Practices1
- Clever0
- 0
- Fork
- Link
- 

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content

- Post
- Cancel

- **Ironslayer**

```
#include <stddef.h>
float sum(int n)
{
  if(n<=1)
    return n;

  return (sum(n-1)+1.0/(3*(n-1)+1));
}

char *series_sum(const size_t n)
{
  char *str=(char*)malloc(n);
  sprintf(str,"%.2f",sum(n));
  return str;
}
```

  - Best Practices1
  - Clever0
  - 0
  - Fork
  - Link
  - 

    Leave feedback...

    - xxxxxxxxxx
    - Your rendered github-flavored markdown will appear here.

    Mark as having spoiler content
    - Post
    - Cancel

- **neuroQuanta**

```
#include <stddef.h>
#define MAX 6

char *series_sum(const size_t n);

char *series_sum(const size_t n) {

  float res = 1.00;
  char *sum = calloc(MAX, sizeof(char));

  (!n) ? (res = 0.00) : res;

  for (int i = 1, d = 4; i < n; ++i, d+=3) res += (1.00/d);

  snprintf(sum, sizeof(sum), "%.2f", res);

  return sum;

}
```

  - Best Practices1
  - Clever0
  - 0
  - Fork
  - Link

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content
- Post
- Cancel

- **whatmename**

```
#include <stddef.h>

char *series_sum(size_t n)
{
    char* r = malloc(0);
    double sum;
    while (n--)
        sum += 1. / (3 * n + 1);
    sprintf(r, "%.2lf", sum);
    return r;
}
```

- Best Practices1
- Clever0
- 0
- Fork
- Link

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content
- Post
- Cancel

- **ImeevMA**

```
#include <stdio.h>
#include <stddef.h>

char *series_sum(const size_t n);

enum
{
    FACTOR = 3,
};
```

```
char *
series_sum(const size_t n)
{
    char *res;
    double sum = 0.0;
    int i;
    for (i = 0; i < n; ++i) {
        int k = FACTOR * i;
        sum += 1.0/(1.0 + (double)k);
    }
    if (asprintf(&res, "%.2f", sum) == -1)
        return -1;
    return res;
}
```

- Best Practices1
- Clever0

- 0
- Fork
- Link

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content

- Post
- Cancel

- **bliss-chris**

```
#include <stddef.h>
#include <stdio.h>

char *series_sum(const size_t n);

char *series_sum(const size_t n)
{
  char * sRet = NULL;
  double res = 0.0;
  double run = 1.0;
  int i;

  sRet = malloc(5);
  if (sRet != NULL)
  {
    for (i=n;i>0;i=(i-1))
    {
      res = res + 1.0/run;
      run += 3.0;
    }
    snprintf(sRet, 5, "%.2e", res);
  }

  return sRet;
}
```

- Best Practices1
- Clever0

- 0
- Fork
- Link

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content
- Post
- Cancel

- **dinglemouse**

```
#include <stddef.h>

char *series_sum(const size_t n) {
  double d = 0;
  for (int i = 0, denom = 1; i < n; i++, denom += 3) d += 1./denom;
  char *buf;
  asprintf(&buf, "%.2f", d);
  return buf;
}
```

- Best Practices1
- Clever0
- 0
- Fork
- Link

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content
- Post
- Cancel

- **GabrielGG2003**

```
char *series_sum(const int  n)
{
  float sum=0;
  for (float i=1;i<n*3;i+=3)
    sum += 1/i;
  char *s = (char *)malloc(3);
  sprintf (s,"%.2f",sum);
  return s;
}
```

- Best Practices1
- Clever0

- 0
- Fork
- Link
- 

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content
- Post
- Cancel

- **snaipasomoh**

```
#include <stddef.h>
#include <stdlib.h>

char *series_sum(int n){
    int a = 1;
    double sum = 0.0;
    for (int i = 0; i < n; i++){
        sum += 1.0/a;
        a += 3;
    }
    char *ans = malloc(sizeof(*ans)*5);
    a = round(sum*100);
    ans[0] = '0' + a/100;
    ans[1] = '.';
    ans[2] = '0' + a/10%10;
    ans[3] = '0' + a%10;
    ans[4] = '\0';
    return ans;
}
```

- Best Practices1
- Clever0
- 0
- Fork
- Link
- 

Leave feedback...

- xxxxxxxxxx
- Your rendered github-flavored markdown will appear here.

Mark as having spoiler content
- Post
- Cancel

Loading more solutions...