

Monitoria de Algoritmos e Estruturas de Dados

2023.2



Vamos falar sobre:

1

Monitores

2

Provas

3

Monitoria

4

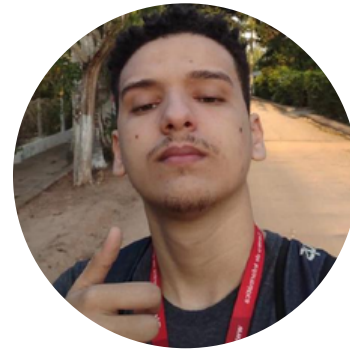
Listas

5

Debug e E/S



Monitores



Andreywid Souza

ayls



Enderson Matos

emmf



Lucas Silva

lls3



Pierre Oriá

pco2



Vladson Marinho

vhrm



Victor Diniz

vpd

Monitorias presenciais:



Dia:
Quintas-Feiras



Horário:
12:00 - 13:00



Local:
Grad 5



Provas:

70% da nota final

2 provas

Listas:

30% da nota final

7 listas

Implementar algoritmos e estruturas vistos em sala (não, não pode usar vector 😞)

Façam as listas!!!

Pontuação Extra:

Possível, não definido exatamente

Durante as monitorias presenciais

Provavelmente faremos com alguns dias de antecedência em relação à prova



Iudex

Árbitro online

Linguagens suportadas:

- C
- C++
- Python
- Java
- Go
- Haskell
- Swift

Critérios

Plágio zero as listas!

C/C++:

- iostream e stdlib, 2 exceções:
 - ~~stdlib::qsort~~
 - ~~stdlib::bsearch~~

Python:

- ~~diccionários~~
- ~~zip~~
- ~~enumerate~~
- ~~append~~
- [...]



Casos-teste

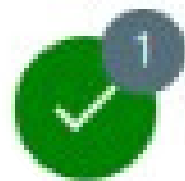
Pontuação ponderada

Apenas a **última**
submissão é válida

Casos ocultos
(grandes)



Vereditos Iudex



AC: Aceito



WA: Wrong Answer



CE: Compilation Error
testem nas suas máquinas antes



MLE: Memory Limit Exceeded



TLE: Tempo Limite Excedido



RE: Runtime Error
erro na execução do código - falha de segmentação, etc



PE: Presentation Error
Formatação errada da saída
(espaços em branco!)

Rodando soluções

- Recomendamos processar a entrada e saída com arquivos, e não testar manualmente

Façam uso dos casos-teste disponíveis no
ludex

Processando entrada e saída:

Sample Input #1

```
1 [[1,2,3]]
2 [5,6]
3
```

Sample Output #1

```
1 [[1,2,3]]
2
```

Sample Input #2

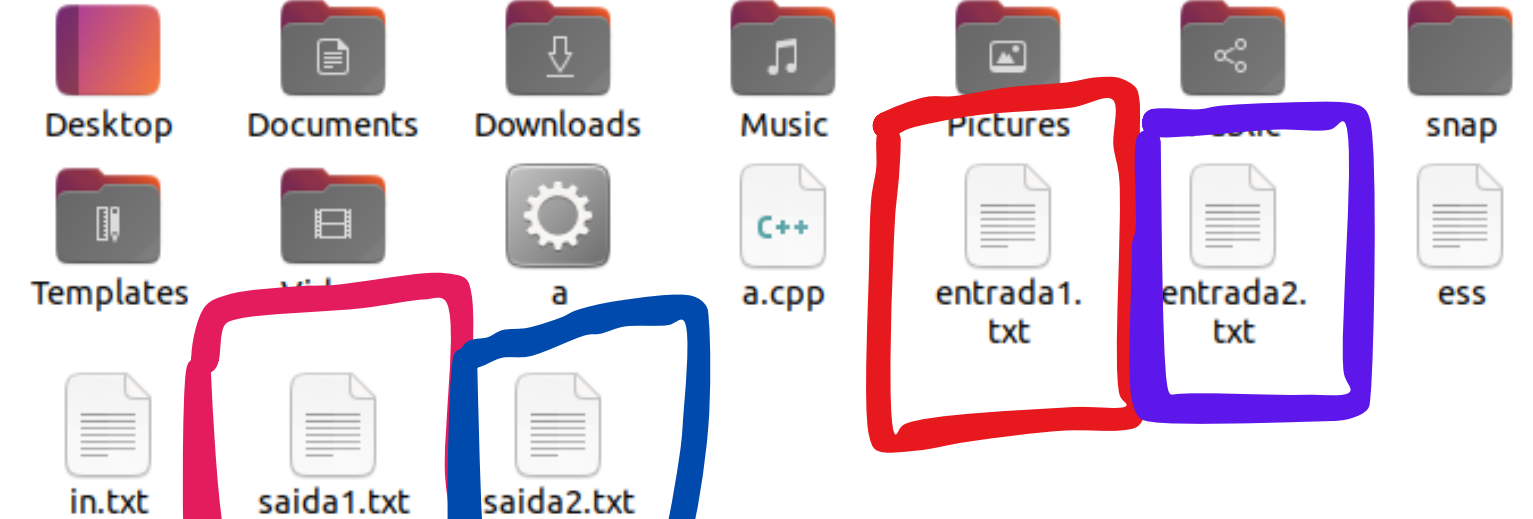
```
[True,False,False]
[True,True,True,True]
[False]
[True]
```

Sample Output #2

```
1 [[True,True,True,True]]
2
```



- Recent
- Starred
- Home
- Documents
- Downloads
- Music
- Pictures
- Videos
- Trash
- + Other Locations



Processando entrada e saída:

1

```
pco2@ug4c05:~$ g++ -o nome_executavel a.cpp
pco2@ug4c05:~$ ./nome_executavel < entrada1.txt > minhasaida1.txt
^C
pco2@ug4c05:~$ diff saida1.txt minhasaida1.txt
pco2@ug4c05:~$
```

1.compilar

Processando entrada e saída:

1

```
pco2@ug4c05:~$ g++ -o nome_executavel a.cpp
```

2

```
pco2@ug4c05:~$ ./nome_executavel < entrada1.txt > minhasaida1.txt
```

```
^C
```

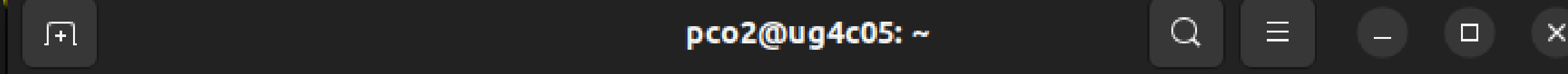
```
pco2@ug4c05:~$ diff saida1.txt minhasaida1.txt
```

```
pco2@ug4c05:~$
```

1.compilar

2.executar, lendo "entrada1" e escrevendo
resultado em "minha saida1"

Processando entrada e saída:



```
pco2@ug4c05: ~  
1 pco2@ug4c05:~$ g++ -o nome_executavel a.cpp  
2 pco2@ug4c05:~$ ./nome_executavel < entrada1.txt > minhasaida1.txt  
^C  
3 pco2@ug4c05:~$ diff saida1.txt minhasaida1.txt  
pco2@ug4c05:~$
```

1.compilar

2.executar, lendo "entrada1" e escrevendo resultado em "minha saida1"

3.verificar diferença entre "saida1" e "minha saida1"

Em Python:

```
1  def main():
2      file1 = open("entrada.txt", "r")
3      file2 = open("saida.txt", "w")
4
5      # lendo - substitui input()
6      inteiro_qualquer = int(file1.readline()[0:-1])
7      string_qualquer = file1.readline()
8
9      resposta = inteiro_qualquer * 10
10     x = 1
11
12     # escrevendo - substitui print()
13     file2.writelines(f"Caso {x}: {resposta}\n")
14
15     # fechando arquivos
16     file1.close()
17     file2.close()
18     # lembrar de usar input, print normal na submissão!
19
20 if __name__ == '__main__':
21     main()
```

Em Python:

depois posso rodar o diff pelo terminal, igual ao exemplo anterior

```
1  def main():
2      file1 = open("entrada.txt", "r")
3      file2 = open("saida.txt", "w")
4
5      # lendo - substitui input()
6      inteiro_qualquer = int(file1.readline()[0:-1])
7      string_qualquer = file1.readline()
8
9      resposta = inteiro_qualquer * 10
10     x = 1
11
12     # escrevendo - substitui print()
13     file2.writelines(f"Caso {x}: {resposta}\n")
14
15     # fechando arquivos
16     file1.close()
17     file2.close()
18     # lembrar de usar input, print normal na submissão!
19
20 if __name__ == '__main__':
21     main()
```

Outras opções

(para entradas pequenas)



Diffchecker

Untitled diff

– 1 removal

1 Alô rivaldo, sai desse lago



+ 1 addition

1 Alô, Rivaldo sai desse lago

Debug:

- **C/C++: usar flags de compilação**
(mensagens de erro mais informativas)
 - **fsanitize=address**
 - **fsanitize=undefined**
 - **Wall, Wextra, Werror**

Sem fsanitize:

```
pco2@ug4c05:~$ g++ -o nome a.cpp
pco2@ug4c05:~$ ./nome < in.txt
Segmentation fault (core dumped)
pco2@ug4c05:~$
```

Com fsanitize:

```
pco2@ug4c05:~$ g++ -o nome a.cpp -fsanitize=address
pco2@ug4c05:~$ ./nome < in.txt
=====
==45259==ERROR: AddressSanitizer: stack-buffer-overflow on address
0 at pc 0x5643bad67462 bp 0x7fff919211b0 sp 0x7fff919211a0
WRITE of size 4 at 0x7fff91921220 thread T0
#0 0x5643bad67461 in main (/home/CIN/pco2/nome+0x2461)
#1 0x7f5133a29d8f in __libc_start_call_main ../sysdeps/nptl/
main.h:58
#2 0x7f5133a29e3f in __libc_start_main_impl ../csu/libc-start
#3 0x5643bad67244 in _start (/home/CIN/pco2/nome+0x2244)

Address 0x7fff91921220 is located in stack of thread T0 at offset
#0 0x5643bad67318 in main (/home/CIN/pco2/nome+0x2318)

This frame has 2 object(s):
  [48, 52) 'n' (line 16)
  [64, 80) 'arr' (line 15) <== Memory access at offset 80 over
ble
HINT: this may be a false positive if your program uses some cus
mechanism, swapcontext or vfork
```

Sem avisos:

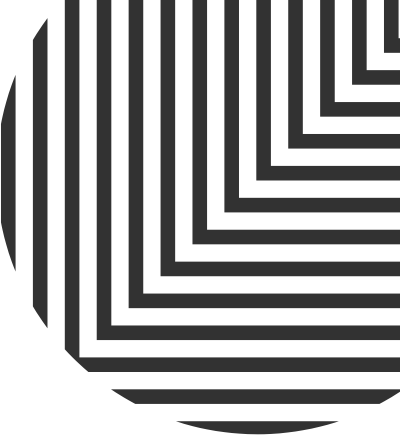
```
pco2@ug4c05:~$ g++ -o nome a.cpp
pco2@ug4c05:~$ ./nome < in.txt
Segmentation fault (core dumped)
pco2@ug4c05:~$
```

Com avisos (Wall):

```
pco2@ug4c05:~$ g++ -o nome a.cpp -Wall
a.cpp: In function 'int main()':
a.cpp:15:9: warning: variable 'arr' set but not used [-Wunu
    15 |     int arr[4];
        |           ^~~
pco2@ug4c05:~$ ./nome < in.txt
Segmentation fault (core dumped)
```

Outras opções:

- **OnlineGDB**
- **Debugger da IDE: VS Code, etc**



FIM