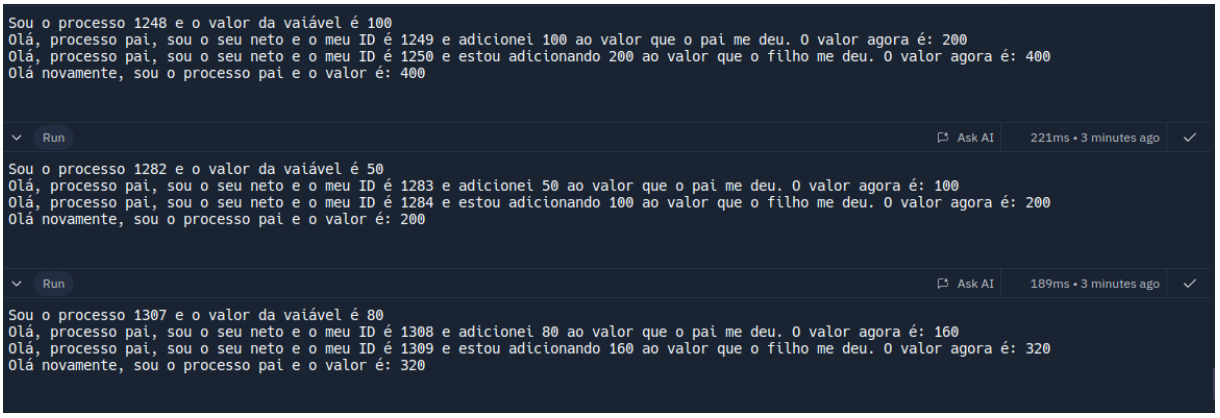
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL IMD
SISTEMAS OPERACIONAIS - IMD0036 -T02

Componente:

- Rafael Araújo de Andrade - 20230036431

Exercício 1:

Nessa questão, eu criei um processo filho e um processo neto para o pai, e quando eu chamava qualquer um dos dois, somava o mesmo valor ao valor inicial, passando como endereço do valor, o local de memória compartilhado entre os 3. Nas seguintes capturas de tela, mostra como estava saindo o valor final:

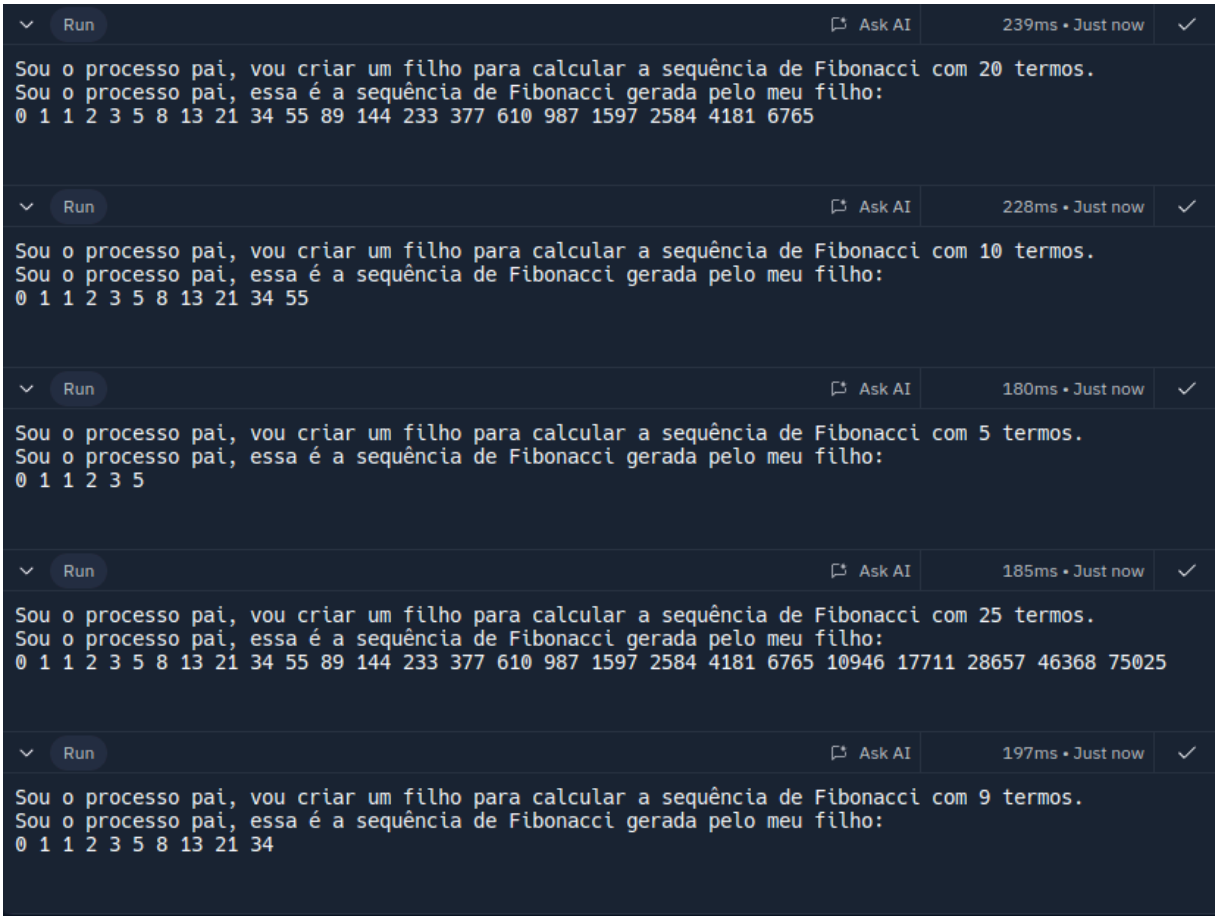


O código da questão pode ser encontrado em:
https://github.com/leafaraujo/SO/blob/bf7c97182a9a4228ec0b39f9ba80dc5484749201/LAB2/questionOne.c

Exercício 2:

No começo, foi um pouco difícil de concatenar todas as saídas esperadas pela questão, mas com um pouco de pesquisa, e com o código resposta colocado pela professora no primeiro laboratório virtual, consegui concatenar todos os valores da sequência de fibonacci, segue a captura de tela com as saídas obtidas:

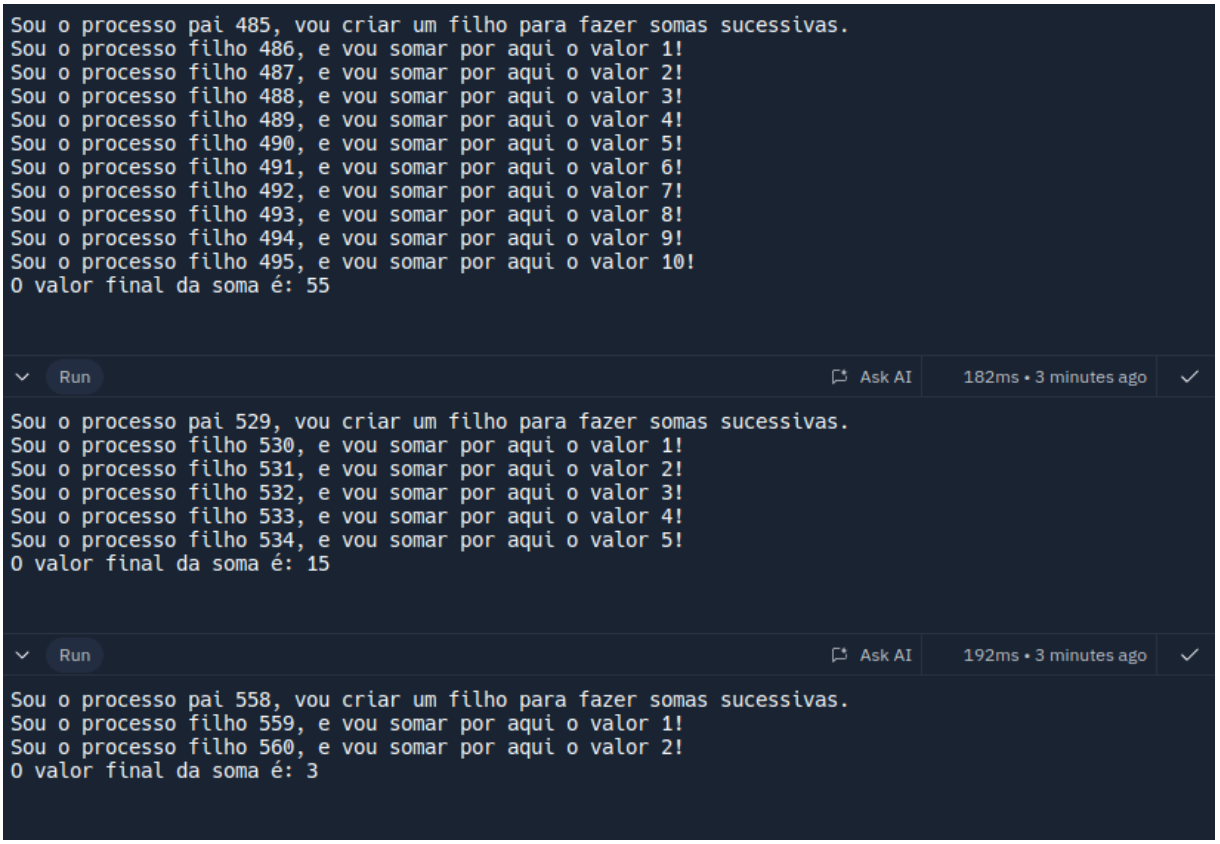
O código da questão pode ser encontrado em:
https://github.com/leafaraujo/SO/blob/c6e9e889db8bbc3bc35c8cfcffc2657b1e50f267/LAB2/questinTwo.c



Exercício 3:

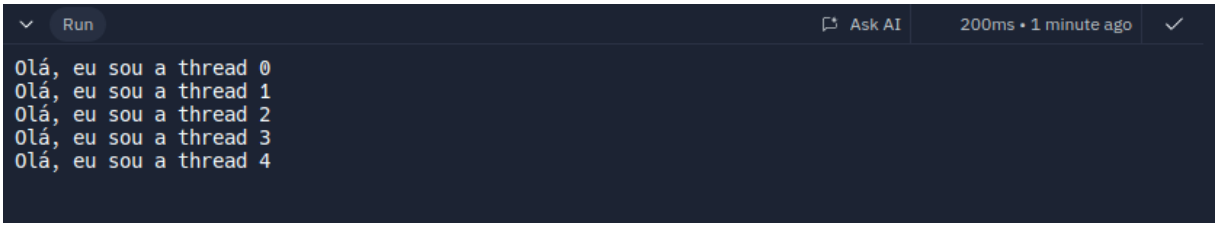
Nesse exercício, a minha maior dificuldade foi em controlar a quantidade de filhos que estavam sendo criados durante o laço for com o valor de n, mas consegui matar o processo filho durante o percorrimento de seu laço, e com isso realizar a soma de forma certa, e com o valor sendo o esperado no final do programa, quando impresso pelo processo 1 na memória compartilhada.

O código da questão pode ser encontrado em:
https://github.com/leafaraujo/SO/blob/acf1f90f300d06dddaf7900b2eae8bed5c132589/LAB2/questionThree.c

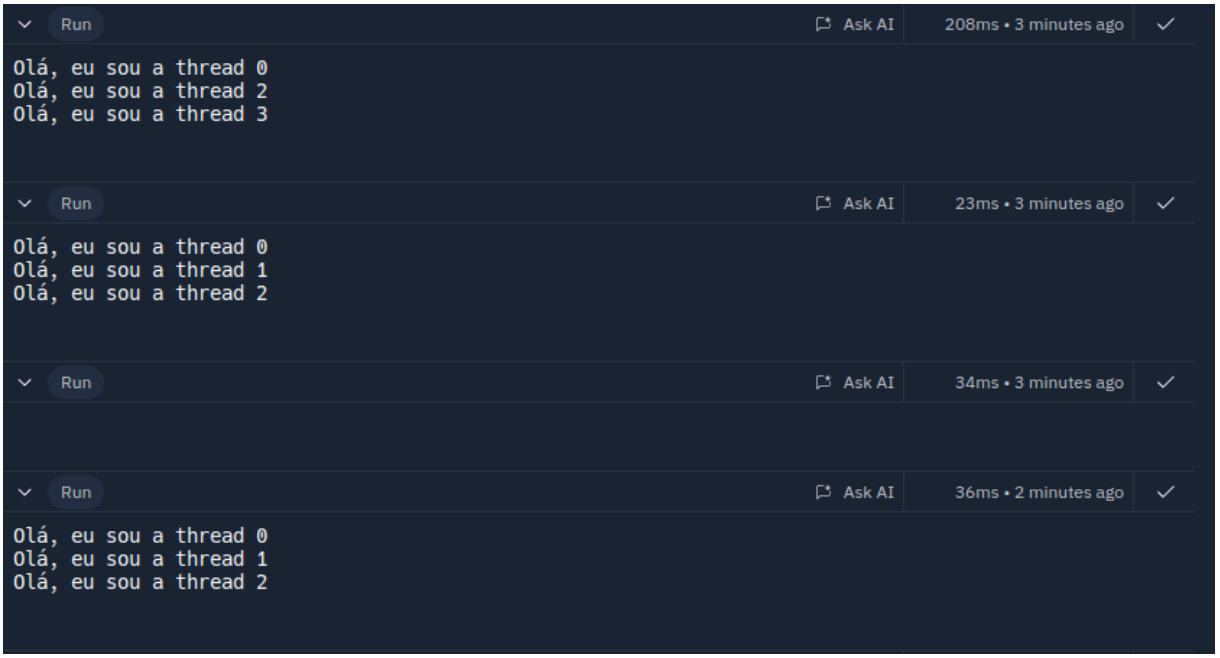


Exercício 4:

Esse é o output do código retirado diretamente do laboratório virtual:



Ao retirar, a linha de código 33 que tem o comando “pthread_join(threads[i],NULL);”, o código apresenta resultados inesperados que podem ser vistos a seguir:



Isso acontece porque a thread principal não está esperando pelo fim da filha, pode-se ver pela terceira execução que inclusive, o programa não conseguiu imprimir nenhuma das threads (o que pode ser erro do compilador, não sei ao certo). Consegui resolver esse problema colocando o programa para dormir logo após a declaração da thread utilizando a função sleep do C, e assim ele esperava que a próxima thread (filha) fosse concluída para encerrar todo o programa.

O código pode ser encontrado no seguinte repositório:
https://github.com/leafaraujo/SO/blob/main/LAB2/questionFour.c