# Trabalho 2 (Fases 4 e 5)

**Edson Ricardo da Costa, Jonatas Van Groll Lemos e Lourenço Souza**

*¹ Sistemas Operacionais - Turma 031*

*² Graduação de Engenharia de Software – Escola Politécnica*

*³ Pontifícia Universidade Católica do Rio Grande do Sul(PUCRS)*

{Edson.Costa, Jonatas.Lemos, Lourenco.Souza} @edu.pucrs.br;

***Resumo.*** *Este pequeno resumo descreve como foi desenvolvido o segundo trabalho da disciplina de Sistemas Operacionais, no 2º semestre de 2021. Mostraremos como foi feita a implementação do trabalho explicando como desenvolvemos o gerenciador de memória e o gerenciador de processos, como se deve executar o programa para poder testar os três casos de teste e mostraremos também o resultado de cada um dos casos de teste desenvolvidos.*

## DESENVOLVIMENTO DA SOLUÇÃO

Com intuito de deixar a solução mais parecida com o código base do professor, para a entrega da fase 4 e 5 nós continuamos utilizando o arquivo "*sistema.java",* nele estão implementados todos os itens da parte 1, 2, 3, 4 e 5. Como a fase 4 e 5 ainda era de implementação pequena, o grupo manteve em um único arquivo o sistema, todo o código desenvolvido segue o padrão sugerido pelo professor, onde temos um código completo e detalhado com comentários. Seguindo essa abordagem, o grupo teve vantagens em desenvolver a nova parte proposta, pois o código já era conhecido pelos integrantes, o que facilitou a implementação do gerenciador de memória e o gerenciador de processos.

A proposta do enunciado da fase 4 era a implementação de um gerenciador de memória para o sistema, esse gerenciador seria responsável por alocar espaço de memória necessário para a execução de um programa, sendo assim, o grupo começou a sua implementação desenvolvendo o gerenciador de memória(Memory Menager). No arquivo do gerenciador nós temos três métodos primordiais, o método responsável por alocar memória, o método responsável por verificar se existe espaço na memória livre e o método responsável por liberar espaço na memória.

O método central do gerenciador de memória é o método de alocar espaço na memória para um programa, pois é com ele que nós separamos o espaço necessário para executar o programa e criamos o processo do programa posteriormente. O método de alocar memória realiza uma série de verificações para permitir o alocamento, caso ocorra algum problema, ele exibe um erro ao usuário e não cria o processo no gerenciador de processos, se ele conseguir alocar o espaço, ele preenche

o espaço de memória livre com os dados do programa que posteriormente será disponibilizado para o gerenciador de processos executar.

Na segunda parte do trabalho, na proposta do enunciado 5 nós tínhamos que implementar um gerenciador de processos, basicamente esse gerenciador de processos teria um gerenciador de memória para alocar memória necessária para executar algum programa. Com a implementação do gerenciador de processos, nós agora deixamos de ter que carregar o programa manualmente na CPU e rodar o sistema diretamente pela CPU. Para executar algum programa, nós primeiro precisamos adicionar esse programa no gerenciador de processo, podemos apenas carregar o programa diretamente no gerenciador de processos, pois o próprio gerenciador se encarrega de carregar o programa no sistema.

Para carregar um programa, o gerenciador de processos utiliza um “Process Control Block” que é um sistema responsável por coordenar e armazenar o programa, nele estão armazenados o programa adicionado, às páginas da memória que esse programa utiliza, os registradores utilizados por esse programa para a execução e um identificador único que é utilizado pelo gerenciador de processos para saber qual processo precisa ser executado.

O gerenciador de processos tem uma lista de processos em andamento, e a cada clock, que é definido por nós, pois criamos uma variável do sistema que armazena o tamanho de tempo do clock, o gerenciador de processos troca o processo em execução para um outro, para assim, ao final de toda a execução do sistema, todos os processos tenham sido finalizado.

Além do mais, o gerenciador de processos (*Process Menager*) conta com seis métodos disponíveis para serem utilizados, temos o método *createProcess*, que cria o processo na memória com auxílio do gerenciador de memória. Temos o método *runProcess*, que executa todos os programas criados na memória, temos também o *runProcessWithId*, caso queiramos executar um programa específico com um identificador criado pelo *process control block*.

Além disso, temos o *killProcessWithId* que mata um processo pelo identificador único do processo, e temos o *killAllProcess*, que mata todos os processos carregados em memória. Fora isso, nós também temos o *schedulerProcess* que é o método central, utilizado para gerenciar qual processo deve ser executado agora, esse método utiliza a variável definida como clock do gerenciador de processos, basicamente esse método é responsável por fazer todo o escalonamento dos programas no sistema. Temos também o *stopProcess*, que é utilizado para finalizar um processo de fato, como o próprio nome pode sugerir.

Para finalizar, sabemos que apenas lendo a descrição feita pelo grupo, podem surgir algumas dúvidas, então sugerimos que caso alguém tenha alguma dúvida de como algo foi implementado, apenas abra o código e olhe, pois lá nós deixamos uma série de comentários explicando o que cada parte do sistema faz na hora de execução.

## RODANDO O PROGRAMA

Para executar o sistema e poder realizar os testes, primeiro você precisa localizar a função “*main*” escrita no arquivo “*Sistema.java*”. É por ela que você executará todos os testes criados pelo grupo, essa função é responsável por controlar o sistema, ou seja, no início da função nós criamos uma nova instância do sistema e logo abaixo chamamos todos os seus respectivos testes.

Além da instância do sistema, nós também temos uma variável chamada “*enableProcessMenagerWarnings*”, essa variável é utilizada como uma flag, ou seja, uma variável de controle para sabermos se queremos ver ou não os avisos que o gerenciador de processos exibe para nós durante a execução do sistema.

Para rodar os novos testes criados, basta seguir o mesmo protocolo utilizado para rodar os testes na versão 1,2 e 3 do sistema. Você precisa escolher qual programa deseja executar e após escolher um dos programas da *main*, basta você descomentar a linha.

Por exemplo, na imagem abaixo nós temos a função *main* com todos os seus testes, nela nós temos uma instância do sistema, temos também a *flag* configurada para *true (assim quando o sistema rodar, o gerenciador de processos irá exibir mensagens para o usuário enquanto o sistema está rodando)* e temos o teste 10 descomentado (“*s.test10()*”), isso significa que no momento que rodarmos a função *main* do sistema o teste 10 será rodado. Caso você deseje rodar algum dos outros testes desenvolvidos pelo grupo, basta comentar a linha do teste 10 e descomentar alguma outra linha de teste.

```java
public static void main(String args[]) {
    //cria uma nova instancia do sistema
    Sistema s = new Sistema();
    s.enableProcessMenagerWarnings = true;

    //test1 - programa que testa fibonnaci
    //s.test1();

    //test2 - programa que testa progminimo
    //s.test2();

    //teste3 - programa que testa fatorial
    //s.test3();

    //test4 - programa que testa interrupções de endereço invalido
    //s.test4();

    //teste5 - programa que testa manipulador de chamada de sistema(trap 1 - input)
    //s.test5();

    //teste6 - programa que testa manipulador de chamada de sistema(trap 2 - output)
    //s.test6();

    //test7 - programa que testa interrupções de intrução invalida
    //s.test7();

    //teste8 - programa que testa interrupções de overflow de operações matematicas
    //s.test8();

    //teste9 - Programa que testa o gerenciador de memória, esse programa apenas carrega a memória com dados "inuteis"
    //s.test9();

    //test10 - programa que testa o gerenciador de processos, esse programa carrega cinco fatoriais diferentes
    s.test10();

    //test11 - programa que testa o gerenciador de processos caso ocorra alguma interrupção que quebre o sistema
    //s.test11();
}
```

Tentamos deixar a função *main* o mais simples possível, para que quando fosse necessário rodar outro programa, o usuário conseguisse realizar a alteração com a maior facilidade possível. Mantivemos o padrão entregue na entrega anterior, das três primeiras fases, ao desenvolver um teste, acima dele existe um comentário com uma breve explicação do que o teste realiza. Caso você queira entender melhor o que o teste faz, basta você procurar no código o nome do teste que você irá facilmente encontrar o código dele desenvolvido.

## RESULTADO DOS TESTES

Este tópico é destinado apenas a apresentação dos resultados obtidos na execução de cada um dos testes desenvolvidos, para compreender melhor o código você precisará abrir o “*Sistema.java*” e analisar como montamos cada um dos códigos.

Sendo assim, abaixo nós temos três testes desenvolvidos para cobrir as fases 4 e 5 do sistema, nos três testes nós falaremos um pouco sobre do que se trata o teste e o que ele testa. Antes de prosseguir para cada um dos casos de teste, primeiro você

precisa entender a estrutura do output do sistema. Quando rodamos o programa, inicialmente você verá uma mensagem avisando que os processos foram adicionados à memória, *true* se foi adicionado com sucesso e *false* caso tenha ocorrido algum erro, se ocorrer erro o sistema avisará qual erro ocorreu.

Outra parte do output é o dump de memória antes de executar o sistema, nele nós conseguimos ver os programas carregados na memória com sucesso, após isso, se a flag do gerenciador de processos estiver configurada para *true*, nós veremos as mensagens que ele exibe quando executa cada um dos processos, depois disso, outro dump de memória é realizado com o output final, com os programas resolvidos e rodados.

## Primeiro Programa de Teste

Neste primeiro programa, o intuito é mostrar o gerenciador de memória em ação, nele nós apenas carregamos quatro programas diferentes que preenchem a memória com dados que não executam nada, sendo assim, no primeiro dump de memória nós temos a memória vazia, sem dados, e no segundo nós temos ela com os programas carregados. Para facilitar a compreensão de que funcionou tudo bem, cada programa está preenchido com um valor diferente.

```
-----------------------------------------------------------------------------
1 - Processo adicionado a memória com sucesso: true
2 - Processo adicionado a memória com sucesso: true
3 - Processo adicionado a memória com sucesso: true
4 - Processo adicionado a memória com sucesso: true
-----------------------------------------------------------------------------
Dump de memória antes de executar todos os programas:
0:   [ ___, -1, -1, -1  ]
1:   [ ___, -1, -1, -1  ]
2:   [ ___, -1, -1, -1  ]
3:   [ ___, -1, -1, -1  ]
4:   [ ___, -1, -1, -1  ]
5:   [ ___, -1, -1, -1  ]
6:   [ ___, -1, -1, -1  ]
7:   [ ___, -1, -1, -1  ]
8:   [ ___, -1, -1, -1  ]
9:   [ ___, -1, -1, -1  ]
10:  [ ___, -1, -1, -1  ]
11:  [ ___, -1, -1, -1  ]
12:  [ ___, -1, -1, -1  ]
13:  [ ___, -1, -1, -1  ]
14:  [ ___, -1, -1, -1  ]
```

*15: [ ___, -1, -1, -1 ]*
*16: [ ___, -1, -1, -1 ]*
*17: [ ___, -1, -1, -1 ]*
*18: [ ___, -1, -1, -1 ]*
*19: [ ___, -1, -1, -1 ]*
*20: [ ___, -1, -1, -1 ]*
*21: [ ___, -1, -1, -1 ]*
*22: [ ___, -1, -1, -1 ]*
*23: [ ___, -1, -1, -1 ]*
*24: [ ___, -1, -1, -1 ]*
*25: [ ___, -1, -1, -1 ]*
*26: [ ___, -1, -1, -1 ]*
*27: [ ___, -1, -1, -1 ]*
*28: [ ___, -1, -1, -1 ]*
*29: [ ___, -1, -1, -1 ]*
*30: [ ___, -1, -1, -1 ]*
*31: [ ___, -1, -1, -1 ]*
*32: [ ___, -1, -1, -1 ]*
*33: [ ___, -1, -1, -1 ]*
*34: [ ___, -1, -1, -1 ]*
*35: [ ___, -1, -1, -1 ]*
*36: [ ___, -1, -1, -1 ]*
*37: [ ___, -1, -1, -1 ]*
*38: [ ___, -1, -1, -1 ]*
*39: [ ___, -1, -1, -1 ]*
*40: [ ___, -1, -1, -1 ]*
*41: [ ___, -1, -1, -1 ]*
*42: [ ___, -1, -1, -1 ]*
*43: [ ___, -1, -1, -1 ]*
*44: [ ___, -1, -1, -1 ]*
*45: [ ___, -1, -1, -1 ]*
*46: [ ___, -1, -1, -1 ]*
*47: [ ___, -1, -1, -1 ]*
*48: [ ___, -1, -1, -1 ]*
*49: [ ___, -1, -1, -1 ]*
*50: [ ___, -1, -1, -1 ]*
*51: [ ___, -1, -1, -1 ]*
*52: [ ___, -1, -1, -1 ]*
*53: [ ___, -1, -1, -1 ]*
*54: [ ___, -1, -1, -1 ]*
*55: [ ___, -1, -1, -1 ]*
*56: [ ___, -1, -1, -1 ]*
*57: [ ___, -1, -1, -1 ]*
*58: [ ___, -1, -1, -1 ]*

*59: [ ___, -1, -1, -1 ]*
*60: [ ___, -1, -1, -1 ]*
*61: [ ___, -1, -1, -1 ]*
*62: [ ___, -1, -1, -1 ]*
*63: [ ___, -1, -1, -1 ]*
*64: [ ___, -1, -1, -1 ]*
*65: [ ___, -1, -1, -1 ]*
*66: [ ___, -1, -1, -1 ]*
*67: [ ___, -1, -1, -1 ]*
*68: [ ___, -1, -1, -1 ]*
*69: [ ___, -1, -1, -1 ]*
*70: [ ___, -1, -1, -1 ]*
*71: [ ___, -1, -1, -1 ]*
*72: [ ___, -1, -1, -1 ]*
*73: [ ___, -1, -1, -1 ]*
*74: [ ___, -1, -1, -1 ]*
*75: [ ___, -1, -1, -1 ]*
*76: [ ___, -1, -1, -1 ]*
*77: [ ___, -1, -1, -1 ]*
*78: [ ___, -1, -1, -1 ]*
*79: [ ___, -1, -1, -1 ]*
*80: [ ___, -1, -1, -1 ]*
*81: [ ___, -1, -1, -1 ]*
*82: [ ___, -1, -1, -1 ]*
*83: [ ___, -1, -1, -1 ]*
*84: [ ___, -1, -1, -1 ]*
*85: [ ___, -1, -1, -1 ]*
*86: [ ___, -1, -1, -1 ]*
*87: [ ___, -1, -1, -1 ]*
*88: [ ___, -1, -1, -1 ]*
*89: [ ___, -1, -1, -1 ]*
*90: [ ___, -1, -1, -1 ]*
*91: [ ___, -1, -1, -1 ]*
*92: [ ___, -1, -1, -1 ]*
*93: [ ___, -1, -1, -1 ]*
*94: [ ___, -1, -1, -1 ]*
*95: [ ___, -1, -1, -1 ]*
*96: [ ___, -1, -1, -1 ]*
*97: [ ___, -1, -1, -1 ]*
*98: [ ___, -1, -1, -1 ]*
*99: [ ___, -1, -1, -1 ]*
*100: [ ___, -1, -1, -1 ]*
*101: [ ___, -1, -1, -1 ]*
*102: [ ___, -1, -1, -1 ]*

*103: [ ___, -1, -1, -1 ]*
*104: [ ___, -1, -1, -1 ]*
*105: [ ___, -1, -1, -1 ]*
*106: [ ___, -1, -1, -1 ]*
*107: [ ___, -1, -1, -1 ]*
*108: [ ___, -1, -1, -1 ]*
*109: [ ___, -1, -1, -1 ]*
*110: [ ___, -1, -1, -1 ]*
*111: [ ___, -1, -1, -1 ]*
*112: [ ___, -1, -1, -1 ]*
*113: [ ___, -1, -1, -1 ]*
*114: [ ___, -1, -1, -1 ]*
*115: [ ___, -1, -1, -1 ]*
*116: [ ___, -1, -1, -1 ]*
*117: [ ___, -1, -1, -1 ]*
*118: [ ___, -1, -1, -1 ]*
*119: [ ___, -1, -1, -1 ]*
*120: [ ___, -1, -1, -1 ]*
*121: [ ___, -1, -1, -1 ]*
*122: [ ___, -1, -1, -1 ]*
*123: [ ___, -1, -1, -1 ]*
*124: [ ___, -1, -1, -1 ]*
*125: [ ___, -1, -1, -1 ]*
*126: [ ___, -1, -1, -1 ]*
*127: [ ___, -1, -1, -1 ]*

---

***Dump de memória depois de executar todos os programas:***

*0: [ DATA, 1, 1, 1 ]*
*1: [ DATA, 1, 1, 1 ]*
*2: [ DATA, 1, 1, 1 ]*
*3: [ DATA, 1, 1, 1 ]*
*4: [ DATA, 1, 1, 1 ]*
*5: [ DATA, 1, 1, 1 ]*
*6: [ DATA, 1, 1, 1 ]*
*7: [ DATA, 1, 1, 1 ]*
*8: [ DATA, 1, 1, 1 ]*
*9: [ DATA, 1, 1, 1 ]*
*10: [ DATA, 1, 1, 1 ]*
*11: [ DATA, 1, 1, 1 ]*
*12: [ DATA, 1, 1, 1 ]*
*13: [ DATA, 1, 1, 1 ]*
*14: [ DATA, 1, 1, 1 ]*
*15: [ DATA, 1, 1, 1 ]*
*16: [ DATA, 1, 1, 1 ]*

*17: [ DATA, 1, 1, 1 ]*
*18: [ DATA, 1, 1, 1 ]*
*19: [ DATA, 1, 1, 1 ]*
*20: [ DATA, 1, 1, 1 ]*
*21: [ DATA, 1, 1, 1 ]*
*22: [ DATA, 1, 1, 1 ]*
*23: [ DATA, 1, 1, 1 ]*
*24: [ DATA, 1, 1, 1 ]*
*25: [ DATA, 1, 1, 1 ]*
*26: [ DATA, 1, 1, 1 ]*
*27: [ DATA, 1, 1, 1 ]*
*28: [ DATA, 1, 1, 1 ]*
*29: [ DATA, 1, 1, 1 ]*
*30: [ DATA, 1, 1, 1 ]*
*31: [ DATA, 1, 1, 1 ]*
*32: [ DATA, 2, 2, 2 ]*
*33: [ DATA, 2, 2, 2 ]*
*34: [ DATA, 2, 2, 2 ]*
*35: [ DATA, 2, 2, 2 ]*
*36: [ DATA, 2, 2, 2 ]*
*37: [ DATA, 2, 2, 2 ]*
*38: [ DATA, 2, 2, 2 ]*
*39: [ DATA, 2, 2, 2 ]*
*40: [ DATA, 2, 2, 2 ]*
*41: [ DATA, 2, 2, 2 ]*
*42: [ DATA, 2, 2, 2 ]*
*43: [ DATA, 2, 2, 2 ]*
*44: [ DATA, 2, 2, 2 ]*
*45: [ DATA, 2, 2, 2 ]*
*46: [ DATA, 2, 2, 2 ]*
*47: [ DATA, 2, 2, 2 ]*
*48: [ DATA, 2, 2, 2 ]*
*49: [ DATA, 2, 2, 2 ]*
*50: [ DATA, 2, 2, 2 ]*
*51: [ DATA, 2, 2, 2 ]*
*52: [ DATA, 2, 2, 2 ]*
*53: [ DATA, 2, 2, 2 ]*
*54: [ DATA, 2, 2, 2 ]*
*55: [ DATA, 2, 2, 2 ]*
*56: [ DATA, 2, 2, 2 ]*
*57: [ DATA, 2, 2, 2 ]*
*58: [ DATA, 2, 2, 2 ]*
*59: [ DATA, 2, 2, 2 ]*
*60: [ DATA, 2, 2, 2 ]*

*61: [ DATA, 2, 2, 2 ]*
*62: [ DATA, 2, 2, 2 ]*
*63: [ DATA, 2, 2, 2 ]*
*64: [ DATA, 3, 3, 3 ]*
*65: [ DATA, 3, 3, 3 ]*
*66: [ DATA, 3, 3, 3 ]*
*67: [ DATA, 3, 3, 3 ]*
*68: [ DATA, 3, 3, 3 ]*
*69: [ DATA, 3, 3, 3 ]*
*70: [ DATA, 3, 3, 3 ]*
*71: [ DATA, 3, 3, 3 ]*
*72: [ DATA, 3, 3, 3 ]*
*73: [ DATA, 3, 3, 3 ]*
*74: [ DATA, 3, 3, 3 ]*
*75: [ DATA, 3, 3, 3 ]*
*76: [ DATA, 3, 3, 3 ]*
*77: [ DATA, 3, 3, 3 ]*
*78: [ DATA, 3, 3, 3 ]*
*79: [ DATA, 3, 3, 3 ]*
*80: [ DATA, 3, 3, 3 ]*
*81: [ DATA, 3, 3, 3 ]*
*82: [ DATA, 3, 3, 3 ]*
*83: [ DATA, 3, 3, 3 ]*
*84: [ DATA, 3, 3, 3 ]*
*85: [ DATA, 3, 3, 3 ]*
*86: [ DATA, 3, 3, 3 ]*
*87: [ DATA, 3, 3, 3 ]*
*88: [ DATA, 3, 3, 3 ]*
*89: [ DATA, 3, 3, 3 ]*
*90: [ DATA, 3, 3, 3 ]*
*91: [ DATA, 3, 3, 3 ]*
*92: [ DATA, 3, 3, 3 ]*
*93: [ DATA, 3, 3, 3 ]*
*94: [ DATA, 3, 3, 3 ]*
*95: [ DATA, 3, 3, 3 ]*
*96: [ DATA, 4, 4, 4 ]*
*97: [ DATA, 4, 4, 4 ]*
*98: [ DATA, 4, 4, 4 ]*
*99: [ DATA, 4, 4, 4 ]*
*100: [ DATA, 4, 4, 4 ]*
*101: [ DATA, 4, 4, 4 ]*
*102: [ DATA, 4, 4, 4 ]*
*103: [ DATA, 4, 4, 4 ]*
*104: [ DATA, 4, 4, 4 ]*

***105: [ DATA, 4, 4, 4 ]***
***106: [ DATA, 4, 4, 4 ]***
***107: [ DATA, 4, 4, 4 ]***
***108: [ DATA, 4, 4, 4 ]***
***109: [ DATA, 4, 4, 4 ]***
***110: [ DATA, 4, 4, 4 ]***
***111: [ DATA, 4, 4, 4 ]***
***112: [ DATA, 4, 4, 4 ]***
***113: [ DATA, 4, 4, 4 ]***
***114: [ DATA, 4, 4, 4 ]***
***115: [ DATA, 4, 4, 4 ]***
***116: [ DATA, 4, 4, 4 ]***
***117: [ DATA, 4, 4, 4 ]***
***118: [ DATA, 4, 4, 4 ]***
***119: [ DATA, 4, 4, 4 ]***
***120: [ DATA, 4, 4, 4 ]***
***121: [ DATA, 4, 4, 4 ]***
***122: [ DATA, 4, 4, 4 ]***
***123: [ DATA, 4, 4, 4 ]***
***124: [ DATA, 4, 4, 4 ]***
***125: [ DATA, 4, 4, 4 ]***
***126: [ DATA, 4, 4, 4 ]***
***127: [ DATA, 4, 4, 4 ]***

## Segundo Programa de Teste

Para o segundo teste, nosso intuito é demonstrar o gerenciador de processos em ação, sendo assim, configuramos a flag dele para true, o que significa que ele irá exibir as mensagens enquanto roda o sistema. Além disso, carregamos o gerenciador de processos com cinco programas de fatorial com valores diferentes, passamos para o programa o fatorial de 6, 5, 4, 3 e 2, sendo assim, no dump final de memória você encontrará os resultados 720, 120, 24, 6 e 2 na respectiva ordem. Marcamos em vermelho no artigo, no dump após a execução do sistema, o valor inserido no fatorial e o valor resultado.

***----------***
***1 - Processo adicionado a memória com sucesso: true***
***2 - Processo adicionado a memória com sucesso: true***
***3 - Processo adicionado a memória com sucesso: true***
***4 - Processo adicionado a memória com sucesso: true***
***5 - Processo adicionado a memória com sucesso: true***
***----------***

***Dump de memória antes de executar todos os programas:***

***0: [ LDI, 0, -1, 6 ]***
***1: [ LDI, 1, -1, 1 ]***
***2: [ LDI, 6, -1, 1 ]***
***3: [ LDI, 7, -1, 8 ]***
***4: [ JMPIE, 7, 0, 0 ]***
***5: [ MULT, 1, 0, -1 ]***
***6: [ SUB, 0, 6, -1 ]***
***7: [ JMP, -1, -1, 4 ]***
***8: [ STD, 1, -1, 10 ]***
***9: [ STOP, -1, -1, -1 ]***
***10: [ DATA, -1, -1, -1 ]***
***11: [ ___, -1, -1, -1 ]***
***12: [ ___, -1, -1, -1 ]***
***13: [ ___, -1, -1, -1 ]***
***14: [ ___, -1, -1, -1 ]***
***15: [ ___, -1, -1, -1 ]***
***16: [ LDI, 0, -1, 5 ]***
***17: [ LDI, 1, -1, 1 ]***
***18: [ LDI, 6, -1, 1 ]***
***19: [ LDI, 7, -1, 8 ]***
***20: [ JMPIE, 7, 0, 0 ]***
***21: [ MULT, 1, 0, -1 ]***
***22: [ SUB, 0, 6, -1 ]***
***23: [ JMP, -1, -1, 4 ]***
***24: [ STD, 1, -1, 10 ]***
***25: [ STOP, -1, -1, -1 ]***
***26: [ DATA, -1, -1, -1 ]***
***27: [ ___, -1, -1, -1 ]***
***28: [ ___, -1, -1, -1 ]***
***29: [ ___, -1, -1, -1 ]***
***30: [ ___, -1, -1, -1 ]***
***31: [ ___, -1, -1, -1 ]***
***32: [ LDI, 0, -1, 4 ]***
***33: [ LDI, 1, -1, 1 ]***
***34: [ LDI, 6, -1, 1 ]***
***35: [ LDI, 7, -1, 8 ]***
***36: [ JMPIE, 7, 0, 0 ]***
***37: [ MULT, 1, 0, -1 ]***
***38: [ SUB, 0, 6, -1 ]***
***39: [ JMP, -1, -1, 4 ]***

*40: [ STD, 1, -1, 10 ]*
*41: [ STOP, -1, -1, -1 ]*
*42: [ DATA, -1, -1, -1 ]*
*43: [ ___, -1, -1, -1 ]*
*44: [ ___, -1, -1, -1 ]*
*45: [ ___, -1, -1, -1 ]*
*46: [ ___, -1, -1, -1 ]*
*47: [ ___, -1, -1, -1 ]*
*48: [ LDI, 0, -1, 3 ]*
*49: [ LDI, 1, -1, 1 ]*
*50: [ LDI, 6, -1, 1 ]*
*51: [ LDI, 7, -1, 8 ]*
*52: [ JMPIE, 7, 0, 0 ]*
*53: [ MULT, 1, 0, -1 ]*
*54: [ SUB, 0, 6, -1 ]*
*55: [ JMP, -1, -1, 4 ]*
*56: [ STD, 1, -1, 10 ]*
*57: [ STOP, -1, -1, -1 ]*
*58: [ DATA, -1, -1, -1 ]*
*59: [ ___, -1, -1, -1 ]*
*60: [ ___, -1, -1, -1 ]*
*61: [ ___, -1, -1, -1 ]*
*62: [ ___, -1, -1, -1 ]*
*63: [ ___, -1, -1, -1 ]*
*64: [ LDI, 0, -1, 2 ]*
*65: [ LDI, 1, -1, 1 ]*
*66: [ LDI, 6, -1, 1 ]*
*67: [ LDI, 7, -1, 8 ]*
*68: [ JMPIE, 7, 0, 0 ]*
*69: [ MULT, 1, 0, -1 ]*
*70: [ SUB, 0, 6, -1 ]*
*71: [ JMP, -1, -1, 4 ]*
*72: [ STD, 1, -1, 10 ]*
*73: [ STOP, -1, -1, -1 ]*
*74: [ DATA, -1, -1, -1 ]*
*75: [ ___, -1, -1, -1 ]*
*76: [ ___, -1, -1, -1 ]*
*77: [ ___, -1, -1, -1 ]*
*78: [ ___, -1, -1, -1 ]*
*79: [ ___, -1, -1, -1 ]*

*----------*

*Programa número 1 está sendo executado*

*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 2 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 3 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 4 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 5 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 1 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 2 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 3 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 4 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 5 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 1 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 2 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 3 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 4 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 5 está sendo executado*

*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 1 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 2 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 3 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 4 está sendo executado*
*Processo Finalizado(interruptionStop)!!!*
*---*
*Programa número 4 foi finalizado.*
*Programa número 5 está sendo executado*
*Processo Finalizado(interruptionStop)!!!*
*---*
*Programa número 5 foi finalizado.*
*Programa número 1 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 2 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 3 está sendo executado*
*Processo Finalizado(interruptionStop)!!!*
*---*
*Programa número 3 foi finalizado.*
*Programa número 1 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*---*
*Programa número 2 está sendo executado*
*Processo Finalizado(interruptionStop)!!!*
*---*
*Programa número 2 foi finalizado.*
*Programa número 1 está sendo executado*
*Processo Finalizado(interruptionStop)!!!*
*---*
*Programa número 1 foi finalizado.*
*---*
*Dump de memória depois de executar todos os programas:*

```
0:   [ LDI, 0, -1, 6  ]
1:   [ LDI, 1, -1, 1  ]
2:   [ LDI, 6, -1, 1  ]
3:   [ LDI, 7, -1, 8  ]
4:   [ JMPIE, 7, 0, 0  ]
5:   [ MULT, 1, 0, -1  ]
6:   [ SUB, 0, 6, -1  ]
7:   [ JMP, -1, -1, 4  ]
8:   [ STD, 1, -1, 10  ]
9:   [ STOP, -1, -1, -1  ]
10:  [ DATA, -1, -1, 720  ]
11:  [ ___, -1, -1, -1  ]
12:  [ ___, -1, -1, -1  ]
13:  [ ___, -1, -1, -1  ]
14:  [ ___, -1, -1, -1  ]
15:  [ ___, -1, -1, -1  ]
16:  [ LDI, 0, -1, 5  ]
17:  [ LDI, 1, -1, 1  ]
18:  [ LDI, 6, -1, 1  ]
19:  [ LDI, 7, -1, 8  ]
20:  [ JMPIE, 7, 0, 0  ]
21:  [ MULT, 1, 0, -1  ]
22:  [ SUB, 0, 6, -1  ]
23:  [ JMP, -1, -1, 4  ]
24:  [ STD, 1, -1, 10  ]
25:  [ STOP, -1, -1, -1  ]
26:  [ DATA, -1, -1, 120  ]
27:  [ ___, -1, -1, -1  ]
28:  [ ___, -1, -1, -1  ]
29:  [ ___, -1, -1, -1  ]
30:  [ ___, -1, -1, -1  ]
31:  [ ___, -1, -1, -1  ]
32:  [ LDI, 0, -1, 4  ]
33:  [ LDI, 1, -1, 1  ]
34:  [ LDI, 6, -1, 1  ]
35:  [ LDI, 7, -1, 8  ]
36:  [ JMPIE, 7, 0, 0  ]
37:  [ MULT, 1, 0, -1  ]
38:  [ SUB, 0, 6, -1  ]
39:  [ JMP, -1, -1, 4  ]
40:  [ STD, 1, -1, 10  ]
41:  [ STOP, -1, -1, -1  ]
```

42: [ DATA, -1, -1, 24 ]
43: [ ___, -1, -1, -1 ]
44: [ ___, -1, -1, -1 ]
45: [ ___, -1, -1, -1 ]
46: [ ___, -1, -1, -1 ]
47: [ ___, -1, -1, -1 ]
48: [ LDI, 0, -1, 3 ]
49: [ LDI, 1, -1, 1 ]
50: [ LDI, 6, -1, 1 ]
51: [ LDI, 7, -1, 8 ]
52: [ JMPIE, 7, 0, 0 ]
53: [ MULT, 1, 0, -1 ]
54: [ SUB, 0, 6, -1 ]
55: [ JMP, -1, -1, 4 ]
56: [ STD, 1, -1, 10 ]
57: [ STOP, -1, -1, -1 ]
58: [ DATA, -1, -1, 6 ]
59: [ ___, -1, -1, -1 ]
60: [ ___, -1, -1, -1 ]
61: [ ___, -1, -1, -1 ]
62: [ ___, -1, -1, -1 ]
63: [ ___, -1, -1, -1 ]
64: [ LDI, 0, -1, 2 ]
65: [ LDI, 1, -1, 1 ]
66: [ LDI, 6, -1, 1 ]
67: [ LDI, 7, -1, 8 ]
68: [ JMPIE, 7, 0, 0 ]
69: [ MULT, 1, 0, -1 ]
70: [ SUB, 0, 6, -1 ]
71: [ JMP, -1, -1, 4 ]
72: [ STD, 1, -1, 10 ]
73: [ STOP, -1, -1, -1 ]
74: [ DATA, -1, -1, 2 ]
75: [ ___, -1, -1, -1 ]
76: [ ___, -1, -1, -1 ]
77: [ ___, -1, -1, -1 ]
78: [ ___, -1, -1, -1 ]
79: [ ___, -1, -1, -1 ]
---

## Terceiro Programa de Teste

Para o terceiro caso de teste, o nosso intuito foi demonstrar que mesmo se inserirmos um programa que irá gerar uma interrupção de sistema o gerenciador de processos irá lidar com isso e executará todos sem quebrar o sistema por completo. Abaixo podemos ver que várias interrupções aparecem nos avisos do gerenciador de processo, mas ele apenas finaliza o processo que gerou essa interrupção e passa para o próximo da lista, ele repete isso até terminar por completo todos os programas adicionados.

```
-------------------------------------------------------------------------------
1 - Processo adicionado a memória com sucesso: true
2 - Processo adicionado a memória com sucesso: true
3 - Processo adicionado a memória com sucesso: true
4 - Processo adicionado a memória com sucesso: true
5 - Processo adicionado a memória com sucesso: true
-------------------------------------------------------------------------------
Dump de memória antes de executar todos os programas:
0:   [ LDI, 1, -1, 50  ]
1:   [ LDI, 7, -1, 7  ]
2:   [ JMPIG, 7, 1, -1  ]
3:   [ LDI, 7, -1, 69  ]
4:   [ STD, 7, -1, 60  ]
5:   [ STOP, 1, -1, 0  ]
6:   [ LDI, 2, -1, 0  ]
7:   [ ADD, 2, 1, -1  ]
8:   [ LDI, 6, -1, 1  ]
9:   [ SUB, 1, 6, -1  ]
10:  [ LDI, 7, -1, 8  ]
11:  [ JMPIG, 7, 1, -1  ]
12:  [ STD, 0, -1, 50  ]
13:  [ STD, 1, -1, 51  ]
14:  [ STD, 2, -1, 52  ]
15:  [ STD, 3, -1, 53  ]
16:  [ STD, 4, -1, 54  ]
17:  [ STD, 5, -1, 55  ]
18:  [ STD, 6, -1, 56  ]
19:  [ STD, 7, -1, 57  ]
20:  [ LDI, 1, -1, 59  ]
21:  [ STD, 1, -1, 1024  ]
```

*22: [ STOP, 1, -1, 0 ]*
*23: [ ___, -1, -1, -1 ]*
*24: [ ___, -1, -1, -1 ]*
*25: [ ___, -1, -1, -1 ]*
*26: [ ___, -1, -1, -1 ]*
*27: [ ___, -1, -1, -1 ]*
*28: [ ___, -1, -1, -1 ]*
*29: [ ___, -1, -1, -1 ]*
*30: [ ___, -1, -1, -1 ]*
*31: [ ___, -1, -1, -1 ]*
*32: [ ___, 8, -1, 1 ]*
*33: [ LDI, 9, -1, 50 ]*
*34: [ TRAP, -1, -1, -1 ]*
*35: [ STOP, 1, -1, 0 ]*
*36: [ DATA, 50, -1, 1 ]*
*37: [ ___, -1, -1, -1 ]*
*38: [ ___, -1, -1, -1 ]*
*39: [ ___, -1, -1, -1 ]*
*40: [ ___, -1, -1, -1 ]*
*41: [ ___, -1, -1, -1 ]*
*42: [ ___, -1, -1, -1 ]*
*43: [ ___, -1, -1, -1 ]*
*44: [ ___, -1, -1, -1 ]*
*45: [ ___, -1, -1, -1 ]*
*46: [ ___, -1, -1, -1 ]*
*47: [ ___, -1, -1, -1 ]*
*48: [ LDI, 0, -1, 2147483647 ]*
*49: [ LDI, 1, -1, 1236 ]*
*50: [ ADD, 0, 1, -1 ]*
*51: [ STD, 0, -1, 8 ]*
*52: [ STD, 1, -1, 9 ]*
*53: [ STOP, 1, -1, 0 ]*
*54: [ DATA, 50, -1, 1 ]*
*55: [ ___, -1, -1, -1 ]*
*56: [ ___, -1, -1, -1 ]*
*57: [ ___, -1, -1, -1 ]*
*58: [ ___, -1, -1, -1 ]*
*59: [ ___, -1, -1, -1 ]*
*60: [ ___, -1, -1, -1 ]*
*61: [ ___, -1, -1, -1 ]*
*62: [ ___, -1, -1, -1 ]*
*63: [ ___, -1, -1, -1 ]*

*64: [ LDI, 8, -1, 1 ]*
*65: [ LDI, 9, -1, 8 ]*
*66: [ TRAP, -1, -1, -1 ]*
*67: [ STOP, 1, -1, 0 ]*
*68: [ ___, -1, -1, -1 ]*
*69: [ ___, -1, -1, -1 ]*
*70: [ ___, -1, -1, -1 ]*
*71: [ ___, -1, -1, -1 ]*
*72: [ ___, -1, -1, -1 ]*
*73: [ ___, -1, -1, -1 ]*
*74: [ ___, -1, -1, -1 ]*
*75: [ ___, -1, -1, -1 ]*
*76: [ ___, -1, -1, -1 ]*
*77: [ ___, -1, -1, -1 ]*
*78: [ ___, -1, -1, -1 ]*
*79: [ ___, -1, -1, -1 ]*
*80: [ LDI, 8, -1, 2 ]*
*81: [ LDI, 9, -1, 8 ]*
*82: [ STD, 9, -1, 8 ]*
*83: [ TRAP, -1, -1, -1 ]*
*84: [ STOP, 1, -1, 0 ]*
*85: [ ___, -1, -1, -1 ]*
*86: [ ___, -1, -1, -1 ]*
*87: [ ___, -1, -1, -1 ]*
*88: [ ___, -1, -1, -1 ]*
*89: [ ___, -1, -1, -1 ]*
*90: [ ___, -1, -1, -1 ]*
*91: [ ___, -1, -1, -1 ]*
*92: [ ___, -1, -1, -1 ]*
*93: [ ___, -1, -1, -1 ]*
*94: [ ___, -1, -1, -1 ]*
*95: [ ___, -1, -1, -1 ]*
*96: [ ___, -1, -1, -1 ]*
*97: [ ___, -1, -1, -1 ]*
*98: [ ___, -1, -1, -1 ]*
*99: [ ___, -1, -1, -1 ]*
*100: [ ___, -1, -1, -1 ]*
*101: [ ___, -1, -1, -1 ]*
*102: [ ___, -1, -1, -1 ]*
*103: [ ___, -1, -1, -1 ]*
*104: [ ___, -1, -1, -1 ]*
*105: [ ___, -1, -1, -1 ]*

*106: [ ___, -1, -1, -1 ]*
*107: [ ___, -1, -1, -1 ]*
*108: [ ___, -1, -1, -1 ]*
*109: [ ___, -1, -1, -1 ]*
*110: [ ___, -1, -1, -1 ]*
*111: [ ___, -1, -1, -1 ]*
*112: [ ___, -1, -1, -1 ]*
*113: [ ___, -1, -1, -1 ]*
*114: [ ___, -1, -1, -1 ]*
*115: [ ___, -1, -1, -1 ]*
*116: [ ___, -1, -1, -1 ]*
*117: [ ___, -1, -1, -1 ]*
*118: [ ___, -1, -1, -1 ]*
*119: [ ___, -1, -1, -1 ]*
*120: [ ___, -1, -1, -1 ]*
*121: [ ___, -1, -1, -1 ]*
*122: [ ___, -1, -1, -1 ]*
*123: [ ___, -1, -1, -1 ]*
*124: [ ___, -1, -1, -1 ]*
*125: [ ___, -1, -1, -1 ]*
*126: [ ___, -1, -1, -1 ]*
*127: [ ___, -1, -1, -1 ]*
*----------*
*Programa número 1 está sendo executado*
*Trocando Processo em Execução(interruptionSchedulerClock)!!!*
*----------*
*Programa número 2 está sendo executado*
*VISHH... Uma interrupção aconteceu enquanto executavamos o programa!!! -->*
*Interrupção: interruptionInvalidInstruction*
*----------*
*Programa número 2 foi finalizado.*
*Programa número 3 está sendo executado*
*VISHH... Uma interrupção aconteceu enquanto executavamos o programa!!! -->*
*Interrupção: interruptionOverflowOperation*
*----------*
*Programa número 3 foi finalizado.*
*Programa número 4 está sendo executado*
*** ------ Chamada de sistema ---------- ***
*Opa... Uma chamada de sistema ocorreu!!! --> | 1 | 8 |*
*--> Por favor digite um valor, apenas inteiros!!!*
*777*
*--> Valor armazenado na posição: 8 --> Valor armazenado: [ DATA, 6, -1, 777 ]*

```
**  ----------------------------------------  **
Processo Finalizado(interruptionStop)!!!
--------------------------------------------------
Programa número 4 foi finalizado.
Programa número 5 está sendo executado
**  ------ Chamada de sistema ----------------  **
Opa... Uma chamada de sistema ocorreu!!! -->  | 2 | 8 |
 --> Output do sistema:  [ DATA, 6, -1, 777  ]
**  ----------------------------------------  **
Trocando Processo em Execução(interruptionSchedulerClock)!!!
--------------------------------------------------
Programa número 1 está sendo executado
VISHH... Uma interrupção aconteceu enquanto executavamos o programa!!! -->
Interrupção: interruptionInvalidInstruction
--------------------------------------------------
Programa número 1 foi finalizado.
Programa número 5 está sendo executado
Processo Finalizado(interruptionStop)!!!
--------------------------------------------------
Programa número 5 foi finalizado.
------------------------------------------------------------------------------------
Dump de memória depois de executar todos os programas:
0:    [ LDI, 1, -1, 50  ]
1:    [ LDI, 7, -1, 7  ]
2:    [ JMPIG, 7, 1, -1  ]
3:    [ LDI, 7, -1, 69  ]
4:    [ STD, 7, -1, 60  ]
5:    [ STOP, 1, -1, 0  ]
6:    [ LDI, 2, -1, 0  ]
7:    [ ADD, 2, 1, -1  ]
8:    [ DATA, 6, -1, 777  ]
9:    [ SUB, 1, 6, -1  ]
10:   [ LDI, 7, -1, 8  ]
11:   [ JMPIG, 7, 1, -1  ]
12:   [ STD, 0, -1, 50  ]
13:   [ STD, 1, -1, 51  ]
14:   [ STD, 2, -1, 52  ]
15:   [ STD, 3, -1, 53  ]
16:   [ STD, 4, -1, 54  ]
17:   [ STD, 5, -1, 55  ]
18:   [ STD, 6, -1, 56  ]
19:   [ STD, 7, -1, 57  ]
```

*20: [ LDI, 1, -1, 59 ]*
*21: [ STD, 1, -1, 1024 ]*
*22: [ STOP, 1, -1, 0 ]*
*23: [ ___, -1, -1, -1 ]*
*24: [ ___, -1, -1, -1 ]*
*25: [ ___, -1, -1, -1 ]*
*26: [ ___, -1, -1, -1 ]*
*27: [ ___, -1, -1, -1 ]*
*28: [ ___, -1, -1, -1 ]*
*29: [ ___, -1, -1, -1 ]*
*30: [ ___, -1, -1, -1 ]*
*31: [ ___, -1, -1, -1 ]*
*32: [ ___, 8, -1, 1 ]*
*33: [ LDI, 9, -1, 50 ]*
*34: [ TRAP, -1, -1, -1 ]*
*35: [ STOP, 1, -1, 0 ]*
*36: [ DATA, 50, -1, 1 ]*
*37: [ ___, -1, -1, -1 ]*
*38: [ ___, -1, -1, -1 ]*
*39: [ ___, -1, -1, -1 ]*
*40: [ ___, -1, -1, -1 ]*
*41: [ ___, -1, -1, -1 ]*
*42: [ ___, -1, -1, -1 ]*
*43: [ ___, -1, -1, -1 ]*
*44: [ ___, -1, -1, -1 ]*
*45: [ ___, -1, -1, -1 ]*
*46: [ ___, -1, -1, -1 ]*
*47: [ ___, -1, -1, -1 ]*
*48: [ LDI, 0, -1, 2147483647 ]*
*49: [ LDI, 1, -1, 1236 ]*
*50: [ ADD, 0, 1, -1 ]*
*51: [ STD, 0, -1, 8 ]*
*52: [ STD, 1, -1, 9 ]*
*53: [ STOP, 1, -1, 0 ]*
*54: [ DATA, 50, -1, 1 ]*
*55: [ ___, -1, -1, -1 ]*
*56: [ ___, -1, -1, -1 ]*
*57: [ ___, -1, -1, -1 ]*
*58: [ ___, -1, -1, -1 ]*
*59: [ ___, -1, -1, -1 ]*
*60: [ ___, -1, -1, -1 ]*
*61: [ ___, -1, -1, -1 ]*

62: [ ___, -1, -1, -1 ]
63: [ ___, -1, -1, -1 ]
64: [ LDI, 8, -1, 1 ]
65: [ LDI, 9, -1, 8 ]
66: [ TRAP, -1, -1, -1 ]
67: [ STOP, 1, -1, 0 ]
68: [ ___, -1, -1, -1 ]
69: [ ___, -1, -1, -1 ]
70: [ ___, -1, -1, -1 ]
71: [ ___, -1, -1, -1 ]
72: [ ___, -1, -1, -1 ]
73: [ ___, -1, -1, -1 ]
74: [ ___, -1, -1, -1 ]
75: [ ___, -1, -1, -1 ]
76: [ ___, -1, -1, -1 ]
77: [ ___, -1, -1, -1 ]
78: [ ___, -1, -1, -1 ]
79: [ ___, -1, -1, -1 ]
80: [ LDI, 8, -1, 2 ]
81: [ LDI, 9, -1, 8 ]
82: [ STD, 9, -1, 8 ]
83: [ TRAP, -1, -1, -1 ]
84: [ STOP, 1, -1, 0 ]
85: [ ___, -1, -1, -1 ]
86: [ ___, -1, -1, -1 ]
87: [ ___, -1, -1, -1 ]
88: [ DATA, -1, -1, 8 ]
89: [ ___, -1, -1, -1 ]
90: [ ___, -1, -1, -1 ]
91: [ ___, -1, -1, -1 ]
92: [ ___, -1, -1, -1 ]
93: [ ___, -1, -1, -1 ]
94: [ ___, -1, -1, -1 ]
95: [ ___, -1, -1, -1 ]
96: [ ___, -1, -1, -1 ]
97: [ ___, -1, -1, -1 ]
98: [ ___, -1, -1, -1 ]
99: [ ___, -1, -1, -1 ]
100: [ ___, -1, -1, -1 ]
101: [ ___, -1, -1, -1 ]
102: [ ___, -1, -1, -1 ]
103: [ ___, -1, -1, -1 ]

*104: [ ___, -1, -1, -1 ]*
*105: [ ___, -1, -1, -1 ]*
*106: [ ___, -1, -1, -1 ]*
*107: [ ___, -1, -1, -1 ]*
*108: [ ___, -1, -1, -1 ]*
*109: [ ___, -1, -1, -1 ]*
*110: [ ___, -1, -1, -1 ]*
*111: [ ___, -1, -1, -1 ]*
*112: [ ___, -1, -1, -1 ]*
*113: [ ___, -1, -1, -1 ]*
*114: [ ___, -1, -1, -1 ]*
*115: [ ___, -1, -1, -1 ]*
*116: [ ___, -1, -1, -1 ]*
*117: [ ___, -1, -1, -1 ]*
*118: [ ___, -1, -1, -1 ]*
*119: [ ___, -1, -1, -1 ]*
*120: [ ___, -1, -1, -1 ]*
*121: [ ___, -1, -1, -1 ]*
*122: [ ___, -1, -1, -1 ]*
*123: [ ___, -1, -1, -1 ]*
*124: [ ___, -1, -1, -1 ]*
*125: [ ___, -1, -1, -1 ]*
*126: [ ___, -1, -1, -1 ]*
*127: [ ___, -1, -1, -1 ]*