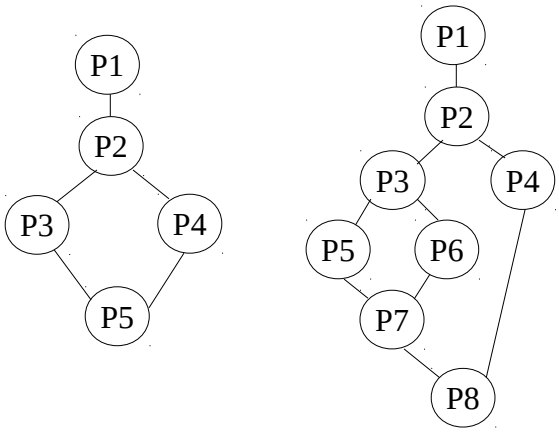
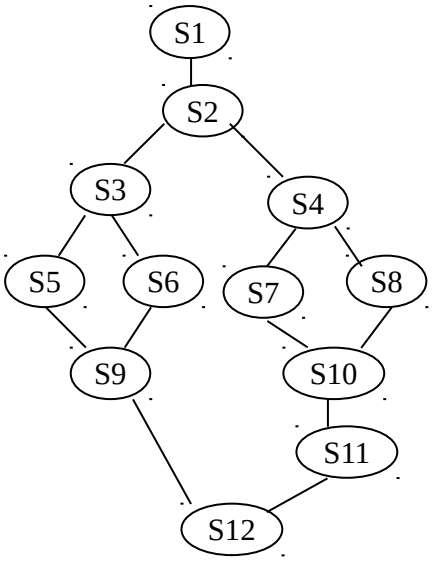
Exercícios de Sistemas Operacionais I
Prática de Programação Concorrente – Fork

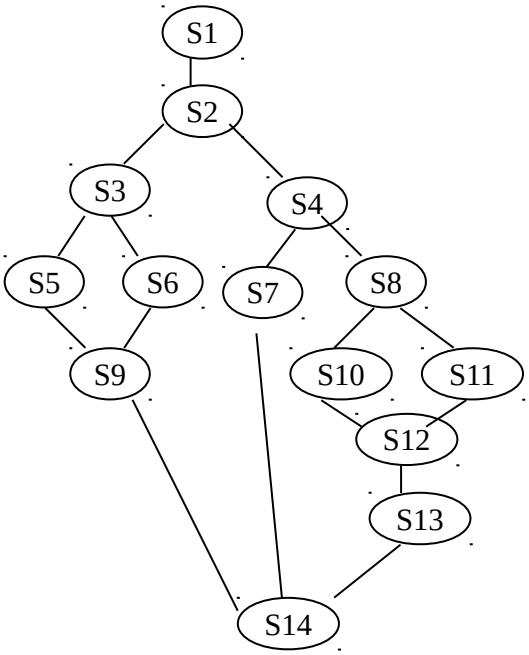
1. Criar um programa que execute os grafos abaixo (um programa para cada). As identificações dos vértices devem ser mostrados na tela. Não é necessário estruturar visualmente como um grafo, apenas manter a ordem de precedência expressa no grafo.



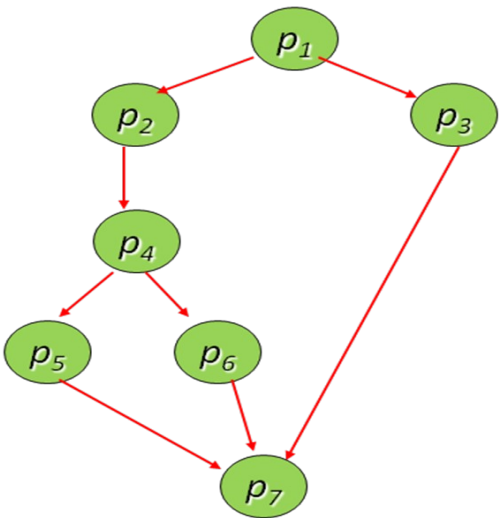
2. Criar um programa que execute o grafo abaixo. As identificações dos vértices devem ser mostrados na tela. Não é necessário estruturar visualmente como um grafo, apenas manter a ordem de precedência expressa no grafo.



3. Criar um programa que execute o grafo a seguir. As identificações dos vértices devem ser mostrados na tela. Não é necessário estruturar visualmente como um grafo, apenas manter a ordem de precedência expressa no grafo.



4. Criar um programa que execute o grafo abaixo. As identificações dos vértices devem ser mostrados na tela. Não é necessário estruturar visualmente como um grafo, apenas manter a ordem de precedência expressa no grafo.



5. Desenhe o grafo de precedência do código a seguir (numerar cada uma das linhas e usar números como vértices do grafo). Em seguida Criar um programa que execute o grafo criado. As identificações dos vértices devem ser mostrados na tela. Não é necessário estruturar visualmente como um grafo, apenas manter a ordem de precedência expressa no grafo.

```
main(){
   int f1; /* Id processo filho 1*/
   int f2; /* Id processo filho 2*/
   printf("Alo do pai\n");
   f1= create_process(codigo_do_filho);
   f2= create_process(codigo_do_filho);
   wait_process( f1);
   printf("Filho 1 morreu\n");
   wait_process( f2);
   printf("Filho 2 morreu\n");
   exit();
}
codigo_do_filho(){
   printf("Alo do filho\n");
   exit();
}
```

6. Desenhe um grafo de precedência do código abaixo (numerar cada uma das linhas e usar números como vértices do grafo). Em seguida Criar um programa que execute o grafo criado. As identificações dos vértices devem ser mostrados na tela. Não é necessário estruturar visualmente como um grafo, apenas manter a ordem de precedência expressa no grafo.:

```
main(){
   int f1, f2, f3;
   printf(“Alo do pai\n”);
   f1=create_process(codigo_do_filho);
   f2=create_process(codigo_do_filho);
   printf(“Filho 1 criado\n”);
   wait_process( f1 );
   printf(“Filho 1 morreu\n”);
   printf(“Filho 2 criado\n”);
   f3=create_process(codigo_do_filho);
   printf(“Filho 3 criado\n”);
   wait_process( f3 );
   printf(“Filho 3 morreu\n”);
   wait_process( f2 );
   printf(“Filho 2 morreu\n”);
   exit();
}
codigo_do_filho(){
   printf(“Alo do filho\n”);
   exit();
}
```

7. Altere o código da questão 5 anterior para que, necessariamente, o processo do filho 2 se inicie somente após o término do processo do filho 1. Reescreva o grafo e o programa e o programa a partir das modificações realizadas.

8. Escrever um programa em linguagem C, com uso da System Call Fork que cria uma árvore de três (3) processos. O processo A faz um fork() criando o processo B. O processo B, por sua vez, faz um fork() criando o processo C. Cada processo deve exibir uma mensagem “Sou o processo pai X e tenho um filho Y” onde X e Y devem ser o número dos processos. Usar a primitiva waitpid() para garantir que a primeira mensagem a aparecer na tela seja do processo A e a última seja do processo C.