1) **Conceitue deadlock**
   A) Situação comumente encontrada em programação concorrente, em que processos têm acesso a um número finito de recursos simultaneamente. Nesse caso, processos ficam liberados infinitamente para usar recursos já alocados.
   B) Situação comumente encontrada em programação concorrente, em que processos concorrentes liberam um número finito de recursos. Nesse caso, processos ficam liberados infinitamente aguardando por recursos não alocados
   C) Situação comumente encontrada em programação concorrente, em que processos concorrentes concorrem a um número finito de recursos. Nesse caso, processos ficam bloqueados infinitamente aguardando por recursos já alocados
   D) Situação comumente encontrada em hardware concorrente em que módulos concorrentes concorrem a um número finito de portas. Nesse caso, módulos ficam liberados infinitamente aguardando por portas não alocadas.

2) **Qual tipo de escalonamento prioriza o período na seleção do próximo processo a ser executado?**
   A) Least slack time
   B) Rate Time
   C) Rate monotonic
   D) Earliest deadline

3) **O que seria o modelo de nomeação indireta?**
   A) Política de comunicação em que o processo origem nomeia uma entidade intermediária para a comunicação com o processo destino.
   B) Política de comunicação em que o processo origem nomeia explicitamente o processo destino.
   C) Política de comunicação em que o processo destino nomeia explicitamente uma porta de recepção
   D) Política de comunicação em que o processo destino nomeia explicitamente o processo destino.

4) **Diferencie os modelos de sincronização síncrono, assíncrono e invocação remota.**
   A) No modelo assíncrono, o processo origem envia a mensagem ao processo destino e fica aguardando por um retorno. No modelo síncrono, o processo origem fica aguardando até a mensagem ser perdida na comunicação. Na invocação remota, o processo origem envia a mensagem para o processo destino e fica aguardando por uma mensagem de reconhecimento do processo destino.
   B) No modelo assíncrono, o processo origem envia a mensagem ao processo destino e aguarda por uma mensagem de reconhecimento. No modelo síncrono, o processo origem envia a mensagem e prossegue. Na invocação remota, o processo origem envia a mensagem para o processo destino e fica aguardando por uma mensagem de reconhecimento do processo destino.
   C) No modelo assíncrono, o processo origem envia a mensagem ao processo destino e prossegue. No modelo síncrono, o processo origem fica aguardando até a mensagem ser recebida pelo processo destino. Na invocação remota, o processo origem envia a

mensagem para o processo destino e fica aguardando por uma mensagem de reconhecimento do processo destino.

D) No modelo assíncrono, o processo origem envia a mensagem ao processo destino e prossegue. No modelo síncrono, o processo origem fica aguardando até a mensagem ser recebida pelo processo destino. Na invocação remota, o processo origem envia a mensagem para o processo destino e prossegue.

5) O que é exclusão mútua?
   A) Restrição de acesso ao hardware
   B) A sincronização requerida para permitir que vários processos acessem uma seção crítica
   C) A sincronização requerida para proteger uma seção crítica é chamada de exclusão mútua.
   D) A sincronização requerida para liberar uma porta de comunicação

6) Selecione a opção que diferencia semáforo binário de variável condicional
   A) Chamada à função wait() no semáforo binário testa o valor do semáforo, e na variável condicional também
   B) Semáforo binário implementa exclusão mútua e variável condicional não.
   C) Chamada à função wait() no semáforo binário não testa o valor do semáforo, e na variável condicional sim.
   D) Chamada à função wait() no semáforo binário testa o valor do semáforo, e na variável condicional não.

7) Selecione a opção que representa uma linha de comando que está presente em uma função de recebimento de mensagem temporizada (timeout).
   A) do { fim = time(NULL); } while((canal[c] == -1) && (difftime(fim, inicio) <= tempo))
   B) do { fim = time(NULL); } while((canal[c] != -1) && (difftime(fim, inicio) <= tempo))
   C) do { fim = time(NULL); } while((canal[c] != -1) && (difftime(fim, inicio) >= tempo))
   D) do { fim = time(NULL); } while((canal[c] != -1) && (difftime(fim, inicio) >= tempo))

8) Selecione a opção que representa uma linha de comando que está presente em uma função de envio sincronizado e temporizado (timeout).
   A) do { fim = time(NULL); } while((canal[c] != -1) && (difftime(fim, inicio) >= tempo))
   B) do { fim = time(NULL); } while((canal[c] == -1) && (difftime(fim, inicio) <= tempo))
   C) do { fim = time(NULL); } while((canal[c] == -1) && (difftime(fim, inicio) >= tempo))
   D) do { fim = time(NULL); } while((canal[c] != -1) && (difftime(fim, inicio) <= tempo))

9) Selecione a opção que representa parte de um escopo de uma medição de tempo transcorrido de alguma ação.

A) do { fim = time(NULL);
} while(ditftime(fim, inicio) <= tempo)

B) inicio = time(NULL)
//ação
fim = time(NULL)
if(fim < inicio)
tempo = time;
printf("%d", tempo);

C) inicio = time(NULL)
//ação
fim = time(NULL);
intervalo = difftime(fim, inicio);

D) inicio = canal[c]
//ação
fim = canal[c]
intervalo = diffcanal(fim, inicio);

10) Selecione a opção que possua uma linha de comando de uma função de recebimento de mensagem por canal e que utilize busy wait loop
A) while(canal[c] != -1);
B) if(canal[c] != -1);
C) if(canal[c] == -1);
D) while(canal[c]) == -1);

11) Com qual valor deve ser inicializado um semáforo de quantidade?
A) Com valor igual a zero.
B) Com a quantidade das threads solicitantes
C) Com valor 1.
D) Com a quantidade de recursos disponíveis.

12) Selecione a opção que contenha uma linha de comando em linguagem C de uma função de recebimento de mensagem não armazenada
A) transfer_memory(canal[c], sizeof(int));
B) memcpy(buf, canal[c], sizeof(int));
C) memcpy(canal[c], sizeof(int));
D) mem_time(buf, sizeof(int));

13) Selecione a opção que conceitua corretamente rendezvous.
A) Estratégia de comunicação em que dois processos ou threads se comunicam periodicamente, permitindo que as duas threads comunicantes enviem suas mensagens em períodos diferentes.

B) Estratégia de comunicação em que dois processos ou threads se comunicam aleatoriamente, permitindo que as duas threads comunicantes enviem suas mensagens em tempos diferentes
C) Estratégia de comunicação em que dois processos ou threads se bloqueiam simultaneamente, permitindo que as duas threads bloqueadas não enviem suas mensagens ao mesmo tempo.
D) Estratégia de comunicação em que dois processos ou threads se comunicam simultaneamente permitindo que as duas threads comunicantes enviem suas mensagens ao mesmo tempo.

14) O que seria o modelo de nomeação assimétrica?
A) Política de comunicação em que os processos origem e destino nomeiam explicitamente um canal para comunicação
B) Política de comunicação em que o processo origem não explicita ou nomeia de quem receberá a mensagem
C) Política de comunicação em que os processos origem e destino nomeiam explicitamente um ente intermediário para comunicação
D) Política de comunicação em que o processo destino não explicita ou nomeia de quem receberá a mensagem

15) Qual função representa a ação de espera seletiva?
A) send_sinc(int *buf, int canal);
B) alt_wait(int quant, int vetor[]);
C) receive(int *buf, int canal);
D) send_assinc(int *but, int canal);

16) Selecione a opção que contenha uma linha de comando em linguagem C de uma função de atraso em segundos que utilize a biblioteca time.h
A) } while(time(NULL) > tempo)
B) } while(time(NULL) <= tempo)
C) } while(difftime(fim, inicio) <= tempo)
D) } while(difftime(fim, inicio) > tempo)

17) Selecione a opção que contenha um escopo de código, em linguagem C, com a função de operação sobre semáforo binário chamada “wait(semáforo). Essa função pode utilizar busy wait loops
A)
```
Wait (int semaforo)
{if(semaforo == 0);
semaforo = semaforo – 1;
}
```
B)
```
Wait (int semaforo)
{do(semáforo == 0);
semaforo = semaforo – 1;
}
```

C) Wait (int semaforo)
{while(semáforo ==0);
semaforo = semaforo – 1;
}

D) Wait (int semaforo)
{do(semáforo ==0);
semaforo = semaforo – 1;
}

18) Selecione a opção que possua uma linha de comando de uma função envio sincronizado por canal (inicializado com -1) e que utilize busy wait loop
   A) while(canal[c] != -1);
   B) if(canal[c] != -1);
   C) if(canal[c] == -1);
   D) while(canal[c]) == -1);

19) Selecione a opção que diferencia semáforo binário de semáforo de quantidade.
   A) Semáforo binário implementa exclusão mutua a apenas um recurso, e o de quantidade, a vários recursos do mesmo tipo.
   B) Semáforo binário implementa exclusão mútua a vários recursos, e o de quantidade a apenas um recurso.
   C) Semáforo binário implementa bloqueio a vários processos, e o de quantidade, a apenas um processo.
   D) Semáforo binário implementa exclusão mútua a vários recursos do mesmo tipo, e o de quantidade, a vários recursos de tipos diferentes.

20) O que seria o modelo de nomeação direta e simétrica?
   A) Política de comunicação em que os processos origem e destino nomeiam explicitamente uma porta de comunicação
   B) Política de comunicação em que os processos origem e destino se nomeiam explicitamente.
   C) Política de comunicação em que os processos origem e destino nomeiam um ente intermediário para comunicação
   D) Política de comunicação em que o processo origem nomeia explicitamente o processo destino e o destino nomeia um canal de comunicação.

21) Elabore um escopo de código com dois processos concorrentes que implementem sincronização através de semáforo binário sobre dois recursos diferentes. Obs: cada processo deve requerer os dois recursos, não pode haver deadlock nessa aplicação

P1	P2
wait(S1);	wait(S1);
wait(S2);	wait(S2);
--	--
signal(S2);	signal(S2);
signal(S1);	signal(S1);