オペレーティングシステム I 令和元年度 後期末試験

(2020.02.05 重村 哲至)　　IE4　　＿＿＿番　氏名＿＿＿＿＿＿　模範解答

1　語句に関する問題

次の文章の空欄に最適な言葉を語群から記号で答えなさい．ただし，空欄 (15) は候補が指定されているので注意すること．(1 点 ×30 問＝ 30 点)

複数のプロセスが同時に並行して実行されるとき，各プロセスが共有変数を勝手に操作すると，プロセスの実行順序によって結果が誤ることがある．このような状況では，共有 **(1)** である共有変数の利用に **(2)** が生じている．プログラム中で **(2)** が生じる可能性のある部分は，**(3)** セクションと呼ばれる．**(2)** が生じないように，**(3)** セクションの実行には相互排除が必要である．相互排除を行うために，**(3)** セクションの入口にある **(4)** セクションでは実行する権利を得る処理を行い，**(3)** セクションの出口にある **(5)** セクションでは実行する権利を返却する処理を行う．**(4)** セクションで権利を得るまでループしながら待つ方式は，**(6)** ウェイティングと呼ばれる．シングルプロセッサ (CPU が 1 個) のシステムでは，**(3)** セクションで **(7)** が発生しないように **(8)** を禁止することで，相互排除を行うことができる．

セマフォは **(9)** とプロセスの **(10)** を管理するデータ型である．セマフォには **(9)** を増やす **(11)** と減らす **(12)** の二つの操作ができる．**(11)** では **(10)** にプロセスがあれば，それの一つを **(13)** にする．**(12)** では **(9)** の値がゼロの場合は，プロセスを **(10)** に追加しプロセスを **(14)** にする．プロセスが **(14)** になるので，CPU を無駄使い **(15)**．

プロセスがメッセージを送信する際，通信相手を指定するためにプロセス番号を用いる方式は **(16)** 指定方式と呼ばれる．メッセージにメッセージの種類を表す **(17)** を付けることができる方式もある．この方式では **(17)** を使って一部のメッセージを選択的に受信することができる．メッセージを送受信する際にプロセスが待ち状態になることがある方式は，**(18)** 方式と呼ばれる．

リーダ・ライタ問題において，**(19)** プロセスは共有データの読み出しだけを，**(20)** プロセスは共有データの読み出しと書き込みの両方を行う．**(19)** プロセスが共有データにかけるロックは **(21)**，**(20)** プロセスが共有データにかけるロックは **(22)** である．

モニタは共有資源の管理機能を持った **(23)** データ型である．モニタ内のメソッド（手続き）は，**(24)** の働きにより，同時には一つしか実行されないことが保証されている．モニタ内で宣言する **(25)** には，signal 操作と wait 操作をすることができる．**(26)** 操作は **(25)** の待ち行列にプロセスを入れる．**(27)** 操作は **(25)** の待ち行列にプロセスがあれば，それの一つをただちに実行する．

資源が排他的に使用され，かつ，**(28)** 不可能な条件下で，**(29)** と **(30)** が同時に起こるとデッドロックが発生する．**(29)** が発生しないようにするには，プロセスが資源を確保する順序に制約を設ける方法がある．**(30)** が発生しないようにするには，プロセスが必要な全ての資源を一度に確保する方法がある．

語群：(あ) signal，(い) wait，(う) P 操作，(え) V 操作，(お) エグジット，(か) エントリー，(き) カウンタ，(く) ガード，(け) クリティカル，(こ) タグ，(さ) ビジー，(し) プリエンプション，(す) ライタ，(せ) リーダ，(そ) 確保待ち，(た) 間接，(ち) 競合，(つ) 共有ロック，(て) 資源，(と) 実行可能，(な) 循環待ち，(に) 条件変数，(ぬ) 排他ロック，(ね) 非同期，(の) 抽象，(は) 直接，(ひ) 同期，(ふ) 横取り，(へ) 待ち行列，(ほ) 待ち状態，(ま) 割り込み，(み) しない (15 の候補)，(む) する (15 の候補)

(1)	(て)	(2)	(ち)	(3)	(け)	(4)	(か)
(5)	(お)	(6)	(さ)	(7)	(し)	(8)	(ま)
(9)	(き)	(10)	(へ)	(11)	(え)	(12)	(う)
(13)	(と)	(14)	(ほ)	(15)	(み)	(16)	(は)
(17)	(こ)	(18)	(ひ)	(19)	(せ)	(20)	(す)
(21)	(つ)	(22)	(ぬ)	(23)	(の)	(24)	(く)
(25)	(に)	(26)	(い)	(27)	(あ)	(28)	(ふ)
(29)	(な)	(30)	(そ)				

オペレーティングシステム I 令和元年度 後期末試験

(2020.02.05 重村 哲至)　IE4　＿＿＿番　氏名＿＿＿模範解答＿＿＿

2 相互排除

1. 次の TeC 風のアセンブリ言語で記述した，二つのスレッドで実行されるプログラムの実行結果について答えなさい。

```
// スレッド 1
        ...
(A)     LD   GO,NUM
(B)     ADD  GO,#1
(C)     ST   GO,NUM
        ...
// スレッド 2
        ...
(a)     LD   GO,NUM
(b)     SUB  GO,#1
(c)     ST   GO,NUM
        ...
// 共有変数
NUM     DC   1          // NUM の初期値=1
```

(a) NUM=1 のとき，A → B → a → b → c → C の順で命令を実行した．実行後の NUM の値を答えなさい。(3 点)

NUM の最終値 ＿2＿

(b) NUM=1 のとき，A → a → B → b → C → c の順で命令を実行した．実行後の NUM の値を答えなさい。(3 点)

NUM の最終値 ＿0＿

(c) このプログラムがシングルプロセッサのシステムで実行されるとき，必ず正しい結果 (NUM = 1) になるようにするにはどうしたらよいか．最も簡単なプログラムの改良方法を説明しなさい．なお，プログラムはカーネルモードで実行されるものとします．(4 点)

(A) と (a) の前の行で割り込みを禁止し，(C) と (c) の次の行で割込み禁止を解除する．

2. 次のような INC 命令と DEC 命令が使用できる場合について答えなさい．

INC（Increment）命令

INC M
- (a) バスをロックする
- (b) $T \leftarrow [M]$
- (c) バスのロックを解除する
- (d) $T \leftarrow T + 1$
- (e) バスをロックする
- (f) $[M] \leftarrow T$
- (g) バスのロックを解除する

DEC（Decrement）命令

DEC M
- (a) バスをロックする
- (b) $T \leftarrow [M]$
- (c) バスのロックを解除する
- (d) $T \leftarrow T - 1$
- (e) バスをロックする
- (f) $[M] \leftarrow T$
- (g) バスのロックを解除する

(a) 1. のプログラムの (A)〜(C) を INC 命令，(a)〜(c) を DEC 命令で置き換えると，シングルプロセッサのシステムでは競合は発生しなくなります．クリティカルセクションの実行の観点を交えて理由を説明しなさい．(5 点)

機械語の途中でプリエンプションは発生しないので，クリティカルセクションである (A)〜(C)，または，(a)〜(c) の処理は，必ず連続して実行される．よって，変数 NUM の更新について競合は発生しない．

(b) (a) と同じことをマルチプロセッサのシステムで行った場合について，以下の二つを答えなさい．(1) 競合は発生しないか，それとも，発生する可能性があるか．(2) なぜそのように考えたか．理由はクリティカルセクションの実行の観点を交えて説明しなさい．(5 点)

(1) 発生する可能性がある．(2)INC 命令と DEC 命令は，途中でバスのロックを解除している．そのため，クリティカルセクションである (a)〜(g) の途中 (d) で別のプロセッサがメモリにアクセスする（割り込む）可能性がある．

オペレーティングシステム I 令和元年度 後期末試験

(2020.02.05 重村 哲至)　　IE4　　＿＿＿番　氏名＿＿＿＿＿＿＿＿ 模範解答 ＿＿＿＿＿＿＿＿

3　セマフォ

1. 次の C 言語風のプログラムにおいて二つの関数 procA() と procB() は，二つのスレッドによって並行実行されます．また，printf() 関数は複数スレッドの環境でも，正常に動作するものとします．以下の問に答えなさい．

```
1  Semaphore S1 = 0;     // 初期値 0 のセマフォ
2  Semaphore S2 = 0;     // 初期値 0 のセマフォ
3  // 以下をスレッド A が実行する
4  void procA() {
5    printf("A-1\n");
6    V( &S2 );
7    P( &S1 );
8    printf("A-2\n");
9    V( &S2 );
10   P( &S1 )
11   printf("A-3\n");
12 }
13 // 以下をスレッド B が実行する
14 void procB() {
15   P( &S2 );
16   printf("B-1\n");
17   V( &S1 );
18   printf("B-2\n");
19   P( &S2 );
20   printf("B-3\n");
21   V( &S1 );
22 }
```

(a) このプログラムが出力する最初の 2 行を順に書きなさい．(3 点)

A-1
B-1

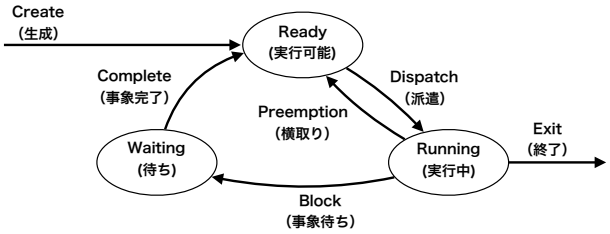
(b) このプログラムが出力する最後の 2 行を順に書きなさい．(3 点)

B-3
A-3

(c) (a)，(b) の 4 行以外の出力行と出力される順序について説明しなさい．(4 点)

A-2，B-2 の 2 行が出力されるが，どちらが先に出力されるか分からない．

2. 次の状態遷移図を参考に答えなさい．



(a) 1. のプログラムにおいて，スレッド A に Block の状態遷移を引き起こす可能性がある行を全て，行番号で答えなさい．(4 点)

7 行，１０行

(b) 1. のプログラムにおいて，スレッド A に Complete の状態遷移を引き起こす可能性がある行を全て，行番号で答えなさい．(4 点)

１７行，２１行

3. 次は C 言語風の仮想言語で記述した，生産者・消費者問題のセマフォによる解です．空欄（_(a)_ など）に適切なセマフォ操作を答えなさい．なお，セマフォの操作の記述は 1. のプログラムを参考にしなさい．(4 点 ×4 箇所=16 点)

```
Data      buffer[N];    // データの格納場所
Semaphore emptySem = N; // セマフォを作る
Semaphore fullSem  = 0; // セマフォを作る
// 生産者スレッド
void producerThread() {
  int in = 0;
  for ( ; ; ) {
    Data d = produce(); // 新しいデータを作る
    _(a)_ ;
    buffer[ in ] = d;
    in = (in + 1) % N;
    _(b)_ ;
  }
}
// 消費者スレッド
void consumerThread() {
  int out = 0;
  for ( ; ; ) {
    _(c)_ ;
    Data d = buffer[ out ];
    out = (out + 1) % N;
    _(d)_ ;
    consume( d );       // データを使用する
  }
}
```

オペレーティングシステムⅠ 令和元年度 後期末試験

(2020.02.05 重村 哲至)　　IE4　　＿＿＿番　氏名＿＿＿＿＿＿＿＿ 模範解答

(a)	P(&emptySem)	(b)	V(&fullSem)
(c)	P(&fullSem)	(d)	V(&emptySem)

4　デッドロック

次のプログラムは，資源を排他的に使用するためにセマフォを使用しています．プログラムをよく読んで問に答えなさい．

```
Resource  r1;            // 資源１
Semaphore s1 = 1;        // 資源１の相互排除用
Resource  r2;            // 資源２
Semaphore s2 = 1;        // 資源２の相互排除用
// スレッド１が実行する関数
void thread1() {
  P(s1);                 // (A)
  資源１を使用する       // (B)
  P(s2);                 // (C)
  資源１と資源２を使用する // (D)
  V(s2);                 // (E)
  V(s1);                 // (F)
}
// スレッド２が実行する関数
void thread2() {
  P(s2);                 // (a)
  P(s1);                 // (b)
  資源１と資源２を使用する // (c)
  V(s1);                 // (d)
  V(s2);                 // (e)
}
```

1. デッドロックが発生し，どちらのスレッドも，それ以上実行できない状態になるまでの実行順の例を，プログラムの行に付けられた (A)〜(F), (a)〜(e) の記号で示しなさい．(4 点)
(例えば「(A) → (a)」のように書く)

(A) → (B) → (a) → (b) → (C)

2. プログラム中の行の順番を入れ替えて，相互排除がされる条件下でもデッドロックが発生しないように改良できます．一組（２行）だけ行の順番を入れ替えることができるとき，入れ替える２行を答えなさい．(4 点)

二つのスレッドで資源の確保順を共通にするれば良いので，(a) 行と (b) 行を逆にする．

3. 次のプログラムは，二つの資源を共通のセマフォで相互排除するように変更したものです．

```
Resource  r1;            // 資源１
Resource  r2;            // 資源２
Semaphore s = 1;         // 資源１，２の相互排除用
// スレッド１が実行する関数
void thread1() {
  P(s);                  // (A)
  資源１を使用する       // (B)
  資源１と資源２を使用する // (C)
  V(s);                  // (D)
}
// スレッド２が実行する関数
void thread2() {
  P(s);                  // (a)
  資源１と資源２を使用する // (b)
  V(s);                  // (c)
}
```

(a) このプログラムはデッドロックの可能性を持っているか答えなさい．また，理由を簡単に（８０文字以下で）説明しなさい．(4 点)

デッドロックは発生しない．セマフォを一つしか使用しないので確保待ちが発生しないからである．

(b) このプログラムの資源の利用効率を変更前のプログラムと比較し簡単に（８０文字以下で）説明しなさい．(4 点)

資源の使用効率は悪くなる．スレッド１が資源１しか使用しない間も，資源２を確保しているからである．