オペレーティングシステムⅠ 令和 4 年度 後期末試験

(2023.02.09 重村 哲至)　　IE4 ____番　氏名 模範解答

注意：以下で「プロセス」と「スレッド」は同じ意味で用いられていることがある.

1　語句に関する問題

次の文章の (1)〜(22) に最適な言葉を語群から記号で答えなさい．また，(a)〜(c) に適切な数値を答えなさい．(1 点 ×25 問＝ 25 点)

ヒント：語群の言葉を全て 1 回使用します.

セマフォは (1) とプロセスの (2) を持つデータ構造である．セマフォに対する (3) 操作は (1) の値を 1 減らす．セマフォに対する (4) 操作は (1) の値を 1 増やす．(1) の値が 0 のセマフォに対して (3) を行ったプロセスは，状態が (5) に変更され (2) に入れられる.

授業では，相互排除問題の解が (a) 個のセマフォを用いて示された．生産者消費者問題（生産者も消費者も単一のもの）の解は (b) 個のセマフォを用いて示された．リーダ・ライタ問題の解は (c) 個のセマフォを用いて示された.

プロセス間通信の代表的な機構として MMU を利用する (6) と，そうではない (7) がある．(7) には相手プロセスを指定して通信する (8) 指定方式と，リンクを指定して通信する (9) 指定方式がある.

(7) はプロセス間の (10) としても働く．通信データはバッファに一時的に保管される．バッファが空のとき，受信のシステムコールがエラーになる方式は (11) 方式，受信のシステムコールがブロックする方式は (12) と呼ばれる.

モニタは資源管理の機能と制約を持った (13) データ型である．モニタの内部には，資源（データ），手続き（プログラム），条件変数などが入っている．(14) の働きにより同時に一つの手続きしか実行できない．条件変数に (15) 操作を行ったプロセスは条件変数の待ち行列に入る．条件変数に (16) 操作を行うと待ち行列のプロセスの一つが (17) に実行される．資源や条件変数はモニタの (18) から直接アクセスすることはできない.

複数のプロセスが互いに待ち合って永遠に処理が進まなくなる現象を (19) と言う．これの原因となる (20) 待ちが発生しないようにするには必要なすべての資源を一度に確保する方法がある．また，(21) 待ちが発生しないようにするには資源の確保順序に制約を設ける方法がある．(22) 問題のように資源の確保順序に制約を設けることができない場合もある.

語群：
(あ) P，(い) signal，(う) V，(え) wait，
(お) Waiting，(か) ガード，(き) カウンタ，
(く) デッドロック，(け) メッセージ通信，
(こ) 外部，(さ) 確保，(し) 間接，
(す) 共有メモリ，(せ) 食事する哲学者，(そ) 循環，
(た) 直ちに，(ち) 直接，(つ) 抽象，(て) 同期，
(と) 同期機構，(な) 非同期，(に) 待ち行列

(1)	(き)	(2)	(に)	(3)	(あ)
(4)	(う)	(5)	(お)	(6)	(す)
(7)	(け)	(8)	(ち)	(9)	(し)
(10)	(と)	(11)	(な)	(12)	(て)
(13)	(つ)	(14)	(か)	(15)	(え)
(16)	(い)	(17)	(た)	(18)	(こ)
(19)	(く)	(20)	(さ)	(21)	(そ)
(22)	(せ)				
(a)	1	(b)	2	(c)	2

オペレーティングシステム I 令和 4 年度 後期末試験

(2023.02.09 重村 哲至)　　IE4 ____番　氏名 模範解答

2　実行順序のトレース

以下の 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．

```
Semaphore S1 = 0;  // 初期値0のセマフォ
Semaphore S2 = 0;  // 初期値0のセマフォ
void procA() {
  P( &S1 );
  printf("A-1\n");
  V( &S1 );
  V( &S1 );
  P( &S2);
  printf("A-2\n");
  V( &S1 );
}
void procB() {
  printf("B-1\n");
  V( &S1 );
  P( &S1 );
  printf("B-2\n");
  P( &S1 );
  printf("B-3\n");
  V( &S2 );
}
```

1. 出力を書きなさい．(10 点)

   B-1
   A-1
   B-2
   B-3
   A-2

2. 終了時の S1 の値を答えなさい．(5 点)

   S1 = 1

3. 終了時の S2 の値を答えなさい．(4 点)

   S2 = 0

3　実行順序の制御

前問と同じ C 言語風のプログラムとスレッド環境において，下に示す順に出力がされ，かつ，セマフォの値が 0 で終了するように，プログラム中「##(?)##」に適切な P 操作，V 操作の記述を解答欄に答えなさい．なお，記述が必要がない場合は解答欄に「×」を記入すること．

```
Semaphore S1 = 0;  // 初期値0のセマフォ
void procA() {
  ##(a)##
  printf("A-1\n");
  ##(b)##
  ##(c)##
  printf("A-2\n");
  ##(d)##
}
void procB() {
  ##(e)##
  printf("B-1\n");
  ##(f)##
  ##(g)##
  printf("B-2\n");
  ##(h)##
}
```

出力：

```
A-1
B-1
A-2
B-2
```

##(a)##	×
##(b)##	V(&S1)
##(c)##	P(&S1)
##(d)##	V(&S1)
##(e)##	P(&S1)
##(f)##	V(&S1)
##(g)##	P(&S1)
##(h)##	×

(2 点 ×8 問=16 点)

オペレーティングシステム I 令和 4 年度 後期末試験

(2023.02.09 重村 哲至)　　IE4 ____番　氏名 模範解答

4　デッドロック

以下の C 言語風のプログラムは，資源１の相互排除にセマフォ S1，資源２の相互排除にセマフォ S2，資源３の相互排除にセマフォ S3 を用います．また，関数 procA，procB，procC は，それぞれ独立したスレッドで並行実行されるものとします．この条件で以下の問に答えなさい．

```
Semaphore S1 = 1;     // 資源1相互排除用
Semaphore S2 = 1;     // 資源2相互排除用
Semaphore S3 = 1;     // 資源3相互排除用
void procA() {
  P( &S2 );           // (A)
  資源２の利用;
  P( &S1 );           // (B)
  資源１・２の利用;
  V( &S1 );           // (C)
  資源２の利用;
  V( &S2 );           // (D)
}
void procB() {
  P( &S1 );           // (E)
  資源１の利用;
  V( &S1 );           // (F)
  P( &S2 );           // (G)
  資源２の利用;
  V( &S2 );           // (H)
}
void procC() {
  P( &S1 );           // (I)
  資源１の利用;
  P( &S2 );           // (J)
  資源１・２の利用;
  V( &S1 );           // (K)
  資源２の利用;
  V( &S2 );           // (L)
}
```

1. procA と同時に実行するとデッドロックが発生する可能性のある関数を全て答えなさい．(5 点)

   procC

2. 1. のデッドロックが発生するまでのセマフォ操作順の例を１つ (A)〜(L) の記号で示しなさい．デッドロックに陥る２つのプロセスが待ち状態になるところまで書くこと．(5 点)

   (A) → (I) → (B) → (J)

3. 効率は悪くなっても構わないので，安全にデッドロックを防ぐには procA をどのように書き換えたらよいか．以下にデッドロックが発生しない procA を書きなさい．（procA 以外は書き換えてはならない．）(10 点)

```
void procA() {
  P( &S1 );
  P( &S2 );
  資源２の利用;
  資源１・２の利用
  資源２の利用;
  V( &S2  );
  V( &S1  );
}
```

オペレーティングシステムⅠ 令和4年度 後期末試験

(2023.02.09 重村 哲至)　　IE4　＿＿番　氏名＿＿模範解答＿＿

5　モニタによるスタックの実装

次は授業で紹介した仮想言語のモニタを用いてスタックを実装した例です．スタックに空きが無い時に push，または，スタックが空の時に pop を実行すると，スレッドは条件変数の待ち行列に入ります．

```
1  monitor Stack {
2    int N;                 // スタックのサイズ
3    int[] stk;             // スタック本体
4    int sp;                // スタックポインタ
5    Condition empty;       // 条件変数
6    Condition full;        // 条件変数
7    Stack(int n) {         // 初期化プログラム
8      N = n;
9      stk = new int[n];
10     sp = 0;
11   }
12   public void push(int n) {
13     if (sp >= N) ##(a)##
14     stk[sp] = n;
15     sp++;
16     empty.signal();
17   }
18   public int pop() {
19     if (sp == 0) ##(b)##
20     sp--;
21     int n = stk[sp];
22     full.signal();
23     return n;
24   }
25 }
```

1. プログラム中「##(?)##」に適切な記述を答えなさい．(5 点 ×2 問=10 点)

##(a)##	full.wait()
##(b)##	empty.wait()

2. 21 行を削除し 23 行を「return stk[sp];」と書き換えても良いか？「良い」か，「良くない」か明確にした上で，理由を１５０文字以内で説明しなさい．(10 点)

良くない
full 条件変数で push 待ちのプロセスが待っていた場合，22 行を実行すると 23 行より先に 14-16 行が実行され，スタックやスタックポインタの状態が変化してしまう．そのため，正しい値を取り出すことができない．21 行で先にスタックからデータを取り出す必要がある．