宿題回答用紙（この用紙でなくても構わないが必要な項目はもれなく書くこと）
宿題番号：課題３，＃１　出題日：11 月 26 日，提出日：12 月 3 日

学籍番号：＿＿＿＿＿＿＿＿＿＿　　氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

講義で説明した全頂点間の最短経路長の計算をドリル形式で理解しよう．

問 1 行列の積のアルゴリズムを理解する

まずは，行列の積を求めるプログラムを理解しよう．ここでは，要素数 n を 5 に固定し，プログラム中で与えられた行列 a に対し，a^2 を求める計算とする．

```
# 準備
n = 5
a = [ [ 1, 2, 3, 4, 5 ],
      [ 0, 0, 0, 0, 0 ],
      [ 5, 4, 3, 2, 1 ],
      [ 1, 1, 1, 0, 0 ],
      [ 0, 0, 0, 1, 0 ],
      [ 0, 0, 0, 0, 1 ] ]

# 答えを入れる配列の用意
ans = Array.new(n){ Array.new(n) }

# 計算
for i in 0..n-1
   for j in 0..n-1
      tmp = 0    ← (*1)
      for k in 0..n-1
         x = a[i][k] * a[k][j]    ← (*2)
         tmp = add(tmp, x)
      end
      ans[i][j] = tmp
   end
end
```

ただし，add(x,y) は x と y の和を求める関数となるように，サブルーチンの形で定義してあるものとする．

(1) i $= 0$, j $= 2$ のとき，(*2) で変数 x に求まる値を示せ．（ヒント：全部で 5 個の値が求まるはず．）

答え：

(2) なぜ，(*1) が無いとどうなるだろうか？

答え：

問 2 総和の計算 ⇒ 最小値の計算

行列の積を計算するときには，関数 add は足し算を計算するサブルーチンだった．たとえば，次のように定義できる．

```
### サブルーチン ############
# 足し算
def add(x, y)
   return(x + y)
end
############################
```

(3) 関数 min(x, y) が x, y の小さい方を返すようにするように，次のサブルーチンの定義を完成させよ．

答え

```
### サブルーチン ############
# 2 つの最小値計算
def min(x, y)
   if
      return(   )
   else
      return(   )
   end
end
############################
```

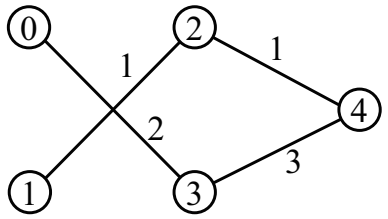
(4) 以下は配列 v の各要素の総和を求めるプログラムだが，add(x, y) を上で定義した min に代えることで，ほぼ同じプログラムで配列中の最小値を求めるプログラムとしても使うことができる．ただし，一箇所，(*1) を変える必要がある．どのように変えればよいだろうか？(ヒント：想定できる最大値を maxnum という変数に入れておく.)

```
# 準備
maxnum = 50 # 想定される最大値
n = 5
v = [ 5, 4, 3, -1, 5 ]

# 計算
tmp = 0  ← (*1)  答え： tmp =
for k in 0..n-1
   x = v[k]
   tmp = add(tmp, x)  ← ここを tmp = min(tmp, x) に代える
end
puts(tmp)
```

問 3 全頂点間最短経路長を計算するアルゴリズム

いよいよ全頂点間最短経路長を計算するアルゴリズムに進もう．ここでの隣接行列は 1 ステップ以下での距離を表わしたものを用いる．つまり，数字は対応する一辺の長さを表している．たとえば，次のようなグラフと隣接行列の関係である．ただし，チェック印は，ここでは -1（意味的には $+\infty$）という数で表わすことにする．



	0	1	2	3	4
0	0	✔	✔	2	✔
1	✔	0	1	✔	✔
2	✔	1	0	✔	1
3	2	✔	✔	0	3
4	✔	✔	1	3	0

こうした隣接行列 a に対して，次のように行列計算と似たような計算の流れで計算すると，2 ステップ以下での距離が求められる．

```
# 準備
maxnum = 50  # 一辺の距離は 50km 未満とする
n = 5
a = [ [  0, -1, -1,  2, -1 ],
      [ -1,  0,  1, -1, -1 ],
      [ -1,  1,  0, -1,  1 ],
      [  2, -1, -1,  0,  3 ],
      [ -1, -1,  1,  3,  0 ] ]

# 配列の準備
a = a の -1 の要素を全部 maxnum に換える
d = a と同じ配列を用意する（答えが求まる配列）
dd = 仮の答えを入れておく配列

# 計算 D = D * A
for i in 0..n-1
   for j in 0..n-1
      tmp =                  ← (*1)
      for k in 0..n-1
         x = d[i][k]         ← (*2)
         tmp = min(tmp, x)
      end
      dd[i][j] = tmp
   end
   d = dd
end
```

(6) 上記の (*1), (*2) を埋めて，2 ステップ以下での距離を配列 d に求めるプログラムを完成させよ．

(7) なぜ，配列 dd を使うのか？

答え

(8) このプログラムに続けて再度，# 計算 D = D * A の部分を繰り返すと，配列 d には何が得られるか？もっと一般的に，次のようなプログラムに直し，計算を m 回繰り返すと，配列 d には何が得られるか？

```
# 計算を m 回繰り返す
t = 1
while t < m
   # 計算 D = D * A
   for i in 0..n-1
      for j in 0..n-1
         tmp =                  ← (*1)
         for k in 0..n-1
            x = d[i][k]         ← (*2)
            tmp = min(tmp, x)
         end
         dd[i][j] = tmp
      end
   end
   d = dd
   t = t + 1
end
```

答え

(9) では，何回繰り返せば目標の D^*_G が求められるだろうか？

答え

さて，もっと速く求める方法は？（ヒント：あの計算技を使おう！）