

.....4.....

Herbert と過ごすテスト期間

59 回生 Faey

今年も、昨年に続いてプログラミング・コンテスト “Imagine Cup 2006” が開催された。このコンテストはいくらかの部門に分かれている。Software Design 部門や Web Development 部門のようなソフトウェア開発の斬新さを競う部門。制作者の表現力を競う ShortFilm 部門。それから、昨年度我々が出場した Visual Gaming 部門。Project Hoshimi と銘打たれたこの競技には、治療用ナノマシンを制御するプログラムを作り、その優秀さを競うという設定がつけられ、後半、運の要素が大きいと批判されながらも、世界各国のプログラマが熱い戦いを繰り広げた。新聞でご存じかもしれないが、私の後輩がこの部門で優勝を果たしている。

さて、今年度も Visual Gaming 部門は開催された。ところが、競技内容は、数点の追加要素を除けば昨年度とほとんど同じ、という内容であった。常に新しきを求める我々にとって、この部門はもはや興味を引くものではなかった。

そこで我々が目をつけたのが、Algorithm 部門であった。昨年度のこの部門は、アルゴリズムの問題をどれだけ多く解けるかを競う、という試験のような代物であった。その上、この問題はすべて英語で記されていた。学生として英語を多少は解するとはいえ、技術英語を読み解きながら英語圏の人間と戦うのは、非常に不利といわざるを得ない。我々は参加を断念した。

ところが、今年度は違っていた。大会側が用意した全く新しい言語に基づく、母国語に依存しない、純粋なアルゴリズム・コンテスト

“Herbert”

それが、今年度の挑戦者に課された課題の名前だった。課題の内容は、「平面を動くロボットを制御するアルゴリズムを最小字数で記述せよ」この内容は、我々の心を動かすに十分な蜜を含んでいた。

こうして、学年末考査も間近に迫った 3 月初旬、我々の戦いは幕を開けた。

新言語 “H”

ロボットを制御するために大会側が提示した言語は、“H” という名を冠していた。言語の基本的な要素 変数・関数という概念の他には、ロボットの移動命令しか持たない。それは平面を移動するという目的に特化された言語であった。

この言語を用いれば、「4 ます前へ進む」という命令は、たった 4 文字で表現できる。

ssss

(4)

おわかりのように、1つの `s` が1ますの前進 (go Straight) を示している。他にも、右折 (turn Right) を示す命令 `r`、左折 (turn Left) を示す命令 `l` が存在し、この三つの組み合わせによってロボットは平面を自由自在に行動する。例えば、「右斜めに2ます移動する」という命令は、(ジグザグに進むことになるが) 以下のようになる。

```
srslsrsl      (8)
```

あるいは、障害物の位置によっては、以下のように動いてもよい。結局のところ、たどり着く位置は同じである。

```
rslsrsls     (8)
```

ところで、この命令が「右斜めに4ます移動する」であった場合、どうすればよいだろう。素直に4回「右斜めに移動する」という命令を書けばよいのだろうか。

```
srslsrslsrslsrsl      (16)
```

コンテストの趣旨を思い出していただきたい。このコンテストは、“最小字数で”アルゴリズムを書くことを求めている。この趣旨に則るならば、「4文字を要する命令」が「4回繰り返され」、16文字もの文字数を消費していることは、非常に無駄であると言わざるを得ない。そこで登場するのが、関数という概念である。

関数

`a` という名前をつけた関数を考えてみよう。“H”では、関数は小文字で書く。この関数は、`srsl`、つまり、斜めに移動する、という機能を備えているとしてほしい。つまり、`a` と書くことで、`srsl` と書いたと同じことである、とするのである。すると、先の16文字は次のように表せる：

```
a:srsl
aaaa      (9)
```

ごらんのように、一行目が関数の定義、即ち、`a` は、`srsl` と同じだという宣言である。この新たに定義した命令 `a` を4回書くことで、`srsl` を4回書いたことと同じと見なすのである。“H”では、`:`、`()`、`+-` の6つの記号を文字と数えないから、この命令は9文字に縮んだことになる。ちなみに、かけ算と割り算は使用できない。

このようにして、アルゴリズムの文字数を縮めることができる。例えば、さらに進む数を増やして「右斜めに9ます移動する」とする。

```
a:srsl
aaaaaaaaa   (14)
```

しかし、`a` を9回繰り返すのは冗長であるから、新たに関数 `b` を定義して、

```
a:srs1
b:aaa
bbb          (12)
```

と書けば、全く同じ命令をより少ない文字数で記すことができる。これらが同じ内容であることは、b を aaa で、a を srs1 で置き換えることによって簡単に示される。

しかし、この方法には限界がある。例えば、「右斜めに 23 ます移動する」という命令を書け、と言われた場合はどうすればよいだろうか。同じ方法に則って、

```
a:srs1
b:aaaaa
aaabbbb      (18)
```

と書くことは可能である。しかし、このままだと進む数やその約数に文字数が大きく左右される。それを防ぐ新たな概念が変数、そして再帰という構造である。

変数と再帰

まず、変数とは、箱である。その箱の中には、数字や命令を放り込んでおける。“H”では、この変数を関数と組み合わせるのみ使用する。例えば、次の命令を与えると、ロボットはどんな行動をとるだろうか。大文字で記された A が変数である。

```
a(A):AA
a(s)          (6)
```

1 行目で、関数 a は A という引数を取ることが宣言された。この場合、関数 a の意味は、

変数 A を受け取り、中に入っている命令を、2 回実行する

というものである。おわかりかもしれないが、2 行目の意味は、a という関数に直進命令 s を与えて呼び出す、というものである。つまり、この命令によって、ロボットは「2 ます直進する」ということになる。ちなみに、この際、A には命令しか入れることができない。その理由は、A に適当な数字を入れて、展開してみればよくわかる。その数字は全く意味をなさない。

この方法を用いれば、例えば「右斜めに 3 ます移動し、さらに左斜めに 3 ます移動する」という命令を、

```
a:srs1
b:slsr
aaabbbb      (16)
```

と書く代わりに、

```
a(A):AAA
a(srs1)a(slsr)  (15)
```

と書くことができる。これで 1 文字縮めることができた。

ちなみに、関数は引数を複数取ることができる。例えば以下のように：

```
a(A,B):AAABBB
a(srs1,s1sr)      (18)
```

この例では望ましくないが、2 つ、あるいは 3 つの引数をとることで縮められる問題も存在した。

さて、やっかいなのが再帰である。これはプログラマにとってもなかなか御しがたい概念で、しかし非常に重要な概念の一つである。幸いにして、この“H”は単純な言語であるから、再帰も用途が限られ、さほど難しくないとされる。では、具体的な例を挙げながら説明してみよう。

まず、再帰とは、文字どおり「再び帰る」、つまり、関数が「自分自身を呼び出す」ことである。これだけでは何のことかわからないだろうから、再帰の例を一つ挙げる。

```
a(A):srsla(A-1)
a(23)      (11)
```

ごらんの通り、関数 a(A) は、その最後で自分自身 a(A-1) を呼び出している。この命令を与えると、ロボットはどう動くだろうか。一つ一つ展開していくと、この命令は次のように働くことがわかる。

```
srsla(22)
srslsrsla(21)
...
srslsrsl...srslsrsla(0)
```

実は、“H”は、関数の引数が 0 以下になると自動的にその関数の実行をやめるという特性を持つ。関数 a はそれぞれ 1 回ずつ「右斜めに進む」という命令を持っていることを考えると、この命令は

「23 回右斜めに進む」

という意味であることがわかる。

おわかりだろうか。この再帰を使うと、実行回数を数字で指定することができるのである。これが、先の疑問の答え。回数が増えたときに、どのように字数を減らすか、である。数字は何桁であろうと 1 文字として扱われるため、単純な「斜めに n 回進む」という命令は、最大 11 文字で書けることが決定された。

それでは、続いてさらに複雑な命令に挑戦してみよう。「右斜めに 13 ます、さらに左斜めに 13 ます進む」という命令を最小字数で書くには、どう書けばよいだろうか。

まず考えられる答えは、

```
a(A):srs1a(A-1)
b(B):srs1b(B-1)
a(13)b(13)      (22)
```

である。ところが、関数は複数の引数を取ることができる。つまり、これを下のように書き直すことができる、ということである。

```
a(A,B):Aa(A,B-1)
a(srs1,13)a(s1sr,13)  (20)
```

さて、この命令はこれ以上縮めることができないのだろうか。それが、できるのである。

堀江理論・応用堀江理論

ここから紹介するのは、我々が発見し命名した数々の特殊技巧の一つである。その他の技巧は、コンテストの問題を解説するうちに紹介する。

堀江理論とは、再帰の性質を利用した繰り返しの技巧である。なぜ堀江と名前がついているのかについては深く問わないでほしいが、関数の働きをみれば、わかる人にはわかるだろう。

堀江理論は、次のような構造を持つ：

```
a(A):sa(A-1)r
a(6)
```

注目していただきたいのは、再帰の後に来ている右折命令 `r` である。この `1` 文字を付け加えることで、どのようなことが起きるかということは、今まで通り展開すればすぐにわかることである。

```
sa(5)r
ssa(4)rr
...
sssssa(0)rrrrrr
```

おわかりだろうか。つまり、再帰の後にたった一文字 `r` (または `1`) をつけることで、ロボットは「くるくるまわる」のである。この場合、6 回右に曲がっているのであるから、ロボットはちょうど来た方向を向くことになる。回数に応じて、最後に向く方向が調節できるのが、この堀江理論の特徴である。

さて、これを応用するのが応用堀江理論である。だが実は、応用といってもたいしたことではない。この堀江理論は、「まわる」ことにこだわらなければ、

指定された回数の命令の後に、同じ回数だけ別の命令を実行できる

という意味にもとれる。これが応用堀江理論である。これを用いて、先の 20 文字の命令を縮めると、次のようになる。

a(A):srs1a(A-1)s1sr
a(13) (15)

これが、右斜めに 13 回進んだ後、左斜めに 13 回進む、ということを意味するのである。この技巧を用いることによって、わずか 15 文字まで縮めることができた。

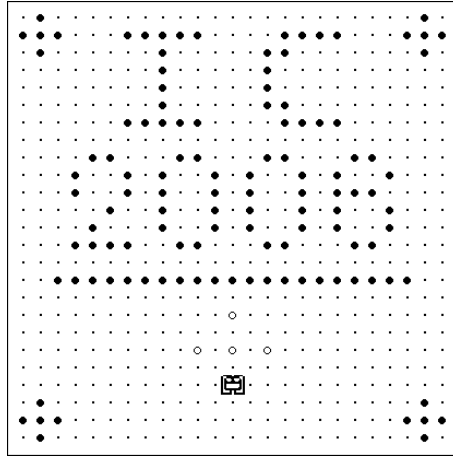
さて、ここからは実際のコンテスト問題の解説に入る。以下に示された解答より短い解答が得られた場合、

<http://npca.my-sv.net/>

にアクセスして、「御意見板」や「恋文投函」に書き込みをどうぞ。

実際の問題解説

Level 1

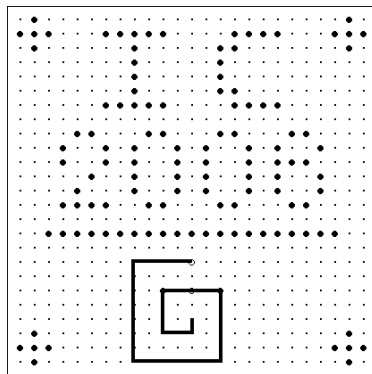


制限字数: 20 文字

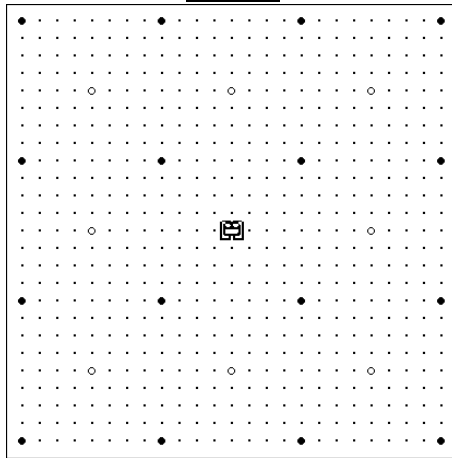
第一問で、問題のルールを説明しておこう。まず、中央下部にある不可思議な形のマークが、ロボットである。つまり、ここがスタート地点であり、このロボットが最初に向いている向きは上である。お察しの通り、このロボットは点の上のみ立ち止まることができる。白い をすべて踏むことによって競技は終了となり、逆に黒い を踏むと、今まで踏んだ がすべてリセットされてしまう。例えばロボットがそこで止まらなくても、すべての を踏んだ瞬間にロボットの動作は終了する。

この競技をするにあたって、最も重要なことは「パターン化できる」軌道を見抜くことである。本問において、踏めばよいのは下方の 4点のみである。さて、これらをもっとも少ない文字数で踏む手順を考えてみてほしい。

もっとも効率の良い軌道を見つければ、この問題は、わずか 8 文字で解くことができる。ヒントは、以下の軌道である。解答は次のページの下部に書いてあるが、挑戦したい方は、ページをめくらず、鉛筆を持ってきてじっくりと考えてみて欲しい。




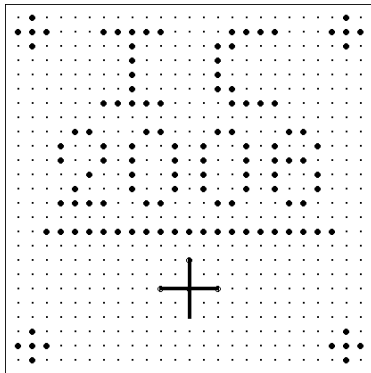
Level 2



制限字数: 20 文字

Level 1 解答



一見すると、中央の  に至り、そこから周囲 3 つの点を取りに行くのがもっとも良いルートであるように見える。その場合の最短命令は、次のようになる：




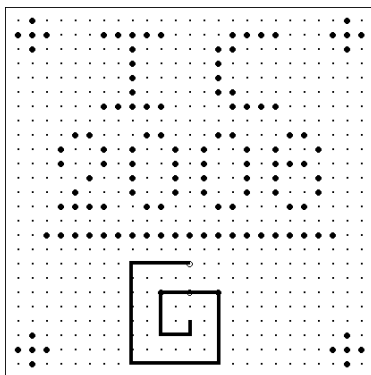
```
a:ssrrssra
```

```
ssa
```

(12)

中央の  まで到達すれば、方向こそ違えど、残りの 3 つの点を取る動作は全く同じである。即ち、「2 ます前に進み、回れ右して 2 ます進む」動作を繰り返せばよいのである。すべての  を取るためには、戻る度に進む方向を 90 度変えてやればよい。

しかし、アルゴリズムの観点から言えば、スタート地点から最後まで全く同じ動作をした方がよい。この場合はどうすればよいだろうか？  回るのである。



```
a(A):rAa(As)
```

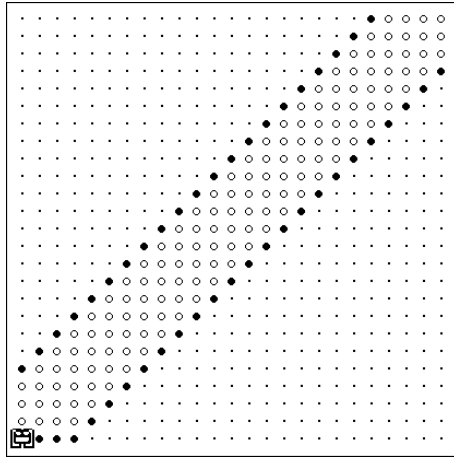
```
a()
```

(8)

最初、関数 a は空の引数を受け取るから、ただ右に曲がるだけの動作をする。その後、自分自身の引数に直進命令 s を次々と付け加えながら、徐々に大きく回っていくのである。

最初の回れ右は狙ったわけではなく、縮めると結果としてそうなることが多い。

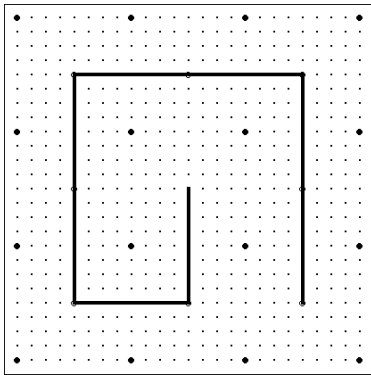
Level 3



制限字数: 13 文字

Level 2 解答

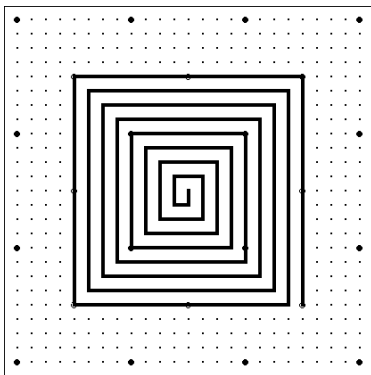
これまた一見すると、以下のように回るのが手っ取り早いように思える。その場合の最短命令は、次のようになる：



```
a:ss
b(B):BBb(aaaaB)
b(r)                                     (15)
```

2回ずつ、同じ距離を進んで右に曲がるという動作を繰り返す。それぞれで進む長さを8長くすれば、図のような軌道を描く。関数 a は、1文字縮めるためのくくりだしである。

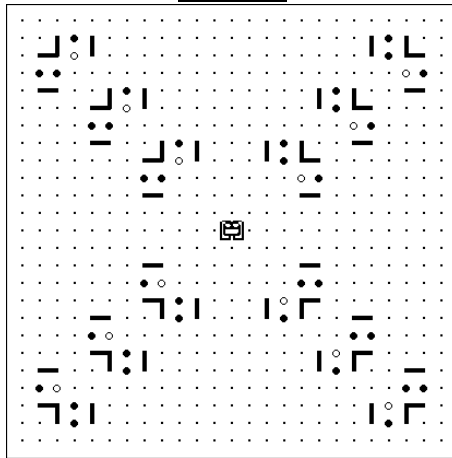
ここで、 の特性を思い出して欲しい。踏めば「今まで踏んだ をリセットする」ということは、まだ一つも を踏んでいない場合、踏んでも平気なのである！



```
a(A):AAa(sA)
a(r)                                     (9)
```

関数 a は、自身の引数の前に前進命令 s を付け加えながら、どんどん回っていく。途中で を踏もうがお構いなしである。

Level 11

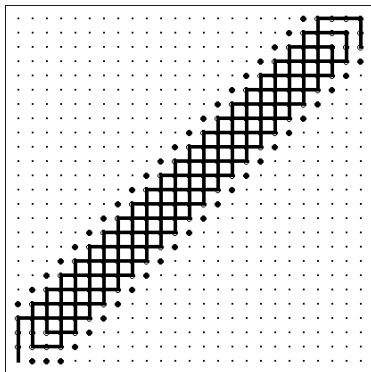


制限字数: 29 文字

Level 3 解答

この問題を解くのは少々やっかいに見える。最初と最初が多少とはいえ不規則に見えるし、斜めに移動しているだけでは制限字数を超えてしまう。

そこで目をつけるのが、制限字数である。上限がわずか 13 文字であるということは、単純なパターンを最初から最後まで、あるいはほとんどの部分を繰り返すだけで解けると考えられる。そのことを念頭に置いて、軌道を見つけると、以下のような軌道になる。



```
f:ssr
a:sfsfffa
a
```

(13)

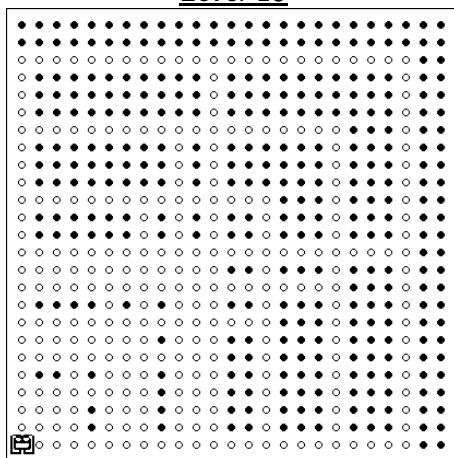
要するに、「3 ます進んで右折」*2、「2 ます進んで右折」*2 の繰り返しである。実は、本問はこれ以下の文字数で解くことはできない。

本問のように、文字数がヒントになって解けてしまう問題はいくつか存在した。Level 11, 17 などは、制限字数がそれぞれ 11 文字、8 文字しかなく、問題を解くというよりはその文字数で考えられるパターンから詰めていった方が早かった。また、Level 48 は高難度問題だが、軌道はすぐに見つかり、16 文字に至るのもすぐである。この問題が Level 48 たるゆえんは、制限字数が 15 文字であることにあるのだが、これもパターンの区切りを変えれば縮めることができる。

高難度問題になると、期間中フォーラム¹などに「どうしても解けない」というスレッドが立ったりしておもしろい。もちろん誰もヒントは漏らさない。

¹theSpoke.net の英語版に、Algorithm 部門のカテゴリがある。そこに苦悩する外国人さんが書き込むのである。

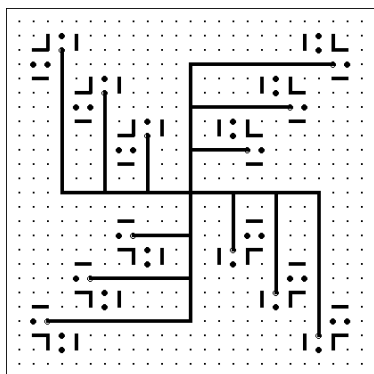
Level 15



制限字数: 24 文字

Level 11 解答

Level が二桁にもなると、だんだんと制限字数に納めるのが難しくなってくる。また、高得点を狙うためには、より少ない字数で解く必要があり、本問のような問題は障害になる。説明のために過程は省き、最短を示す。



$$a(A, N) : \text{AsArsAllAa}(A, N-1)a(\text{sssA}, 4) \\ a(r, 4) \quad (25)$$

一つ を取ったら原点に戻り、同じ距離にある 4 つを取った時点でより外側の 4 つを取りに行く。これは非常に最適化されているが、ここまでしなくても解ける。関数の引数は必ずしも同じ種類でなくともよい。

注意して欲しいのは、ここで「自分自身を呼ぶ」つまり再帰を、自分で 2 回行っていることである。より左にある再帰の呼び出しでは、残り回数を 1 回減らして呼んでいるが、右側の呼び出しでは定数 4 を回数として指定している。つまり、これは無限ループであり、実際はすべて を踏んでもどんどん外側の を取りに行こうとする。一種の応用堀江理論だと言ってもよいかもしれない。

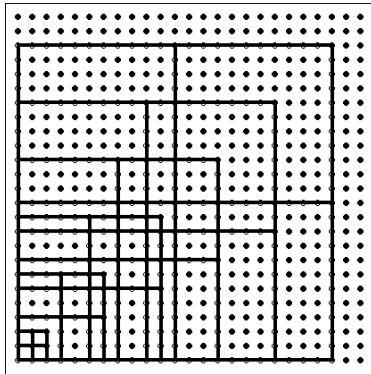
また、関数の中で、All という文字列があるが、これは A の中に入っている r を打ち消して、さらに左折しようというものである。ちなみに、関数の最初の呼び出しで r を入れるのはよくあることである。入れなければこうなる：

$$a(A, N) : \text{ArsArrsAlAra}(A, N-1)a(\text{sssA}, 4) \\ a(, 4) \quad (26)$$

これだけで数文字縮められたりする。

Level 15 解答

この問題は、少々軌道が見つけづらいかもしれない。「田」という形をした、大きさの違う軌道がいくつも重なっている、と言えばわかるだろうか？ぐちゃぐちゃしているが、以下が軌道である。



$$\begin{aligned} a(A, N) &: AA1AAa(A, N-1)a(ssA, 4) \\ a(sr, 4) \end{aligned} \quad (21)$$

Level 11 のような形式で解くとなると、このような命令になる。ホッチキスの針のような形をした軌道を 4 回繰り返すことで、「田」を表現している。あとはサイズを大きくしながら繰り返すだけである。r のくりだしを外すと軌道がよくわかる。

ところが、これには非常にすばらしい解が存在する。この軌道、よく見ると同じ命令が繰り返されることが非常に多い。関数 a が「田」一つを表すように関数を組み替えると、

$$\begin{aligned} a(A) &: ArAArArArAArArArAArArArAArAra(ssA) \\ a(s) \end{aligned}$$

となる。a の中身を共通部分に分解すると、

$$\begin{aligned} &ArAArArArAArArArAArArArAArAr \\ &= ArAArArArAArAr * 2 \\ &= ArAArAr * 2 * 2 \\ &= ArA ArA r * 2 * 2 \end{aligned}$$

そこで、その命令を 2 回ずつセットにして考えると、以下のようなコードが生まれ出される。

$$\begin{aligned} f(A) &: AA \\ a(A) &: f(f(f(ArA)r))a(ssA) \\ a(s) \end{aligned} \quad (19)$$

よりハイレベルな問題について

紙面の都合上、問題の解説はわずか 5 問、それも前半の簡単な問題ばかりとなってしまったが、後半の難問も、軌道さえ見つければだいたいがこれらの技法で解ける。しかし、最後の 10 問ほどは、3 つの変数を使い、階差的な軌道を表すという超高難度技法を使わなければ解けないものばかりで、Level 46 などは、全参加者のうちたった 2 人しか解を導くことができなかった。

ヒントは、3 引数 A,B,C をとり、B が A に、C が B に影響されるような関数である。興味のある方は、theSpoke.net などへ行って、競技者有志が公開している問題をダウンロードし挑戦してみたい。