★オリジナル問題集の回答と解説

　このオリジナル問題集は「論理的思考力」を身につけることを目指して作成しています。私は「論理的思考力」とは目的を達成するために最適な手段を選べるようになるための思考力だと考えています。

　この後の回答と解説は問題の答えだけではなく、どういう考え方を意識すれば「論理的思考力」を身につけられるのかエンジニア目線で書いていますので、参考にしていただければ幸いです m(_ _)m

レベル：★☆☆☆☆

出題意図：

レベル 1 は目的(スタートからゴールへ行くこと)を達成するためにどうすればいいのか考える問題です。どんな手段(プログラム)でもかまわないので、まずは目的を達成することを意識しましょう。

回答：

割愛

レベル：★★☆☆☆

出題意図：

レベル 2 はレベル 1 に簡単な条件(通過点)を加えた問題です。レベル 1 に比べて目的を達成する手段に制限が加わるので、目的を達成する手段が少し難しくなります。

回答：

割愛

レベル：★★★☆☆

出題意図：

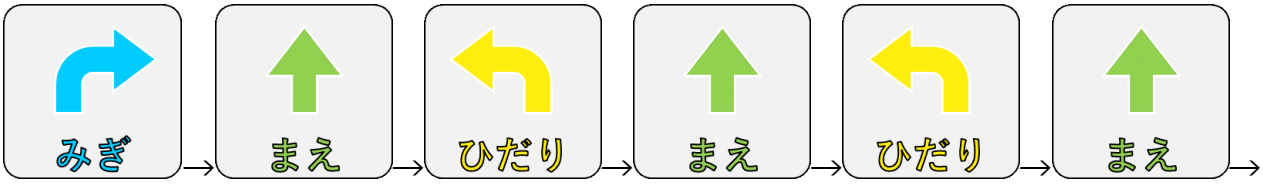
レベル 3 はレベル 2 にちょっと難しい条件(制限事項)を加えた問題です。レベル 2 に比べて目的を達成する手段にさらに制限が加わるので、目的を達成する手段が少し難しくなります。

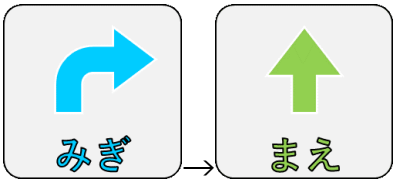
準備（じゅんび）：　ボットリー（ぼっとりー）を　1　に　うえ　むきで　おこう！

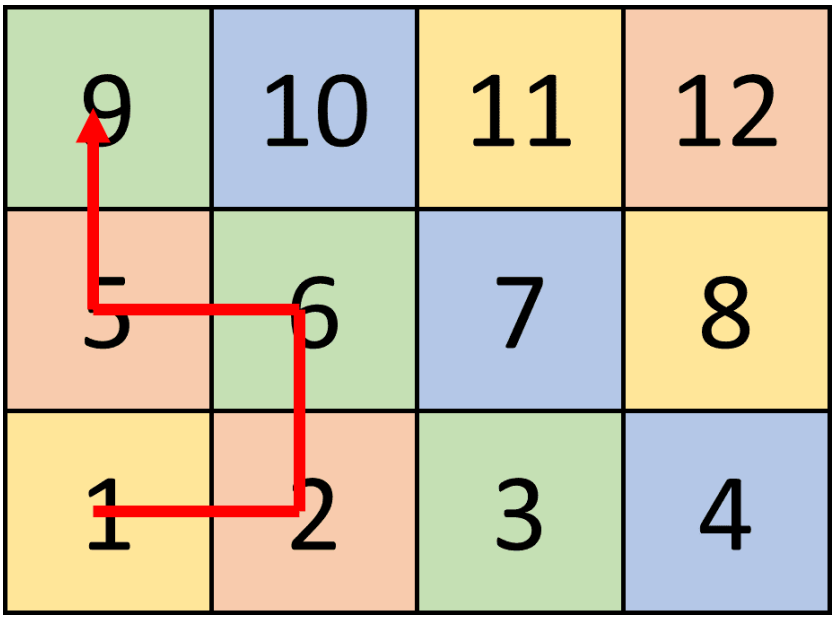
3-1：　1　から　6　をとおって　9　へいこう！

ただし、9　には　うえ　むきで　ゴール（ごーる）しよう！

回答例：







解説：

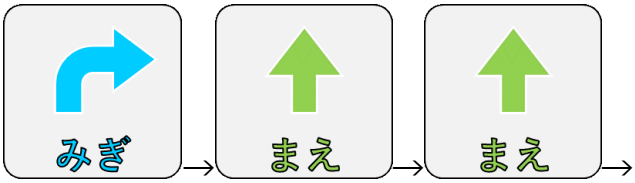
この問題はスタート地点とゴール地点でのボットリーの向きが決まっていることがポイントです。条件(スタート時の向き、通過点、ゴール時の向き)を満たせばどのようなルートを通ってもOKです。

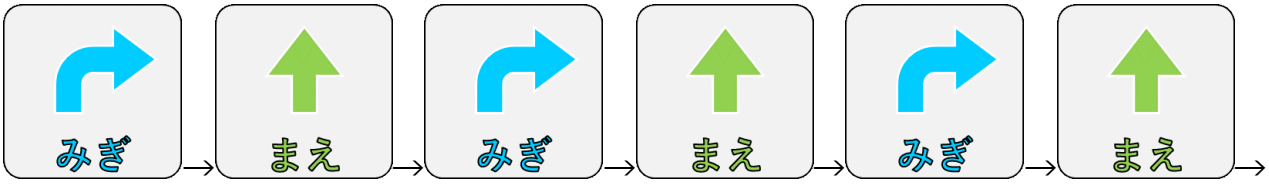
目的を達成する手段は複数あるので、その中から最適な手段を選択できる様になることは重要です。多くの場合は少ない手間で目的を達成できることが良いとされています。

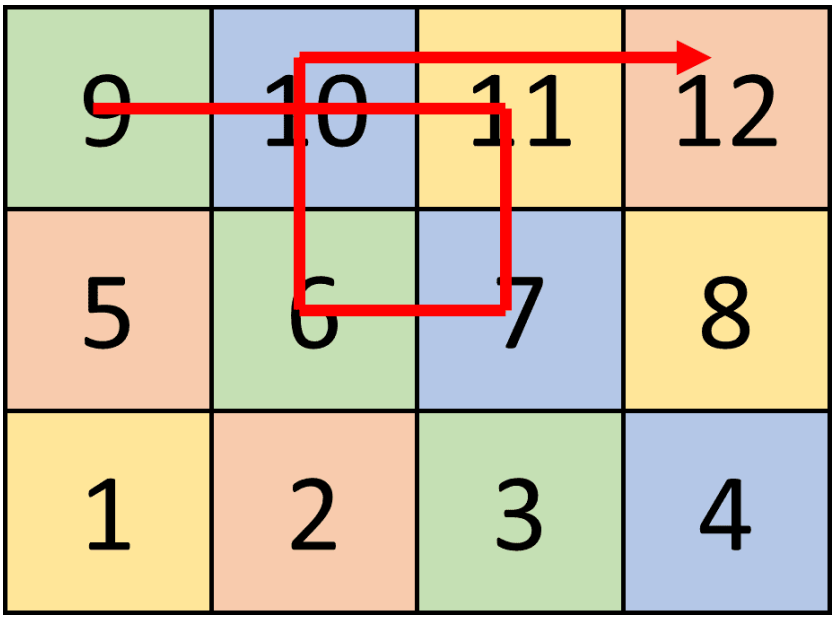
3-2： 9 から 7 をとおって 12 へいこう！

ただし、ひだり をつかわないで いこう！

回答例：







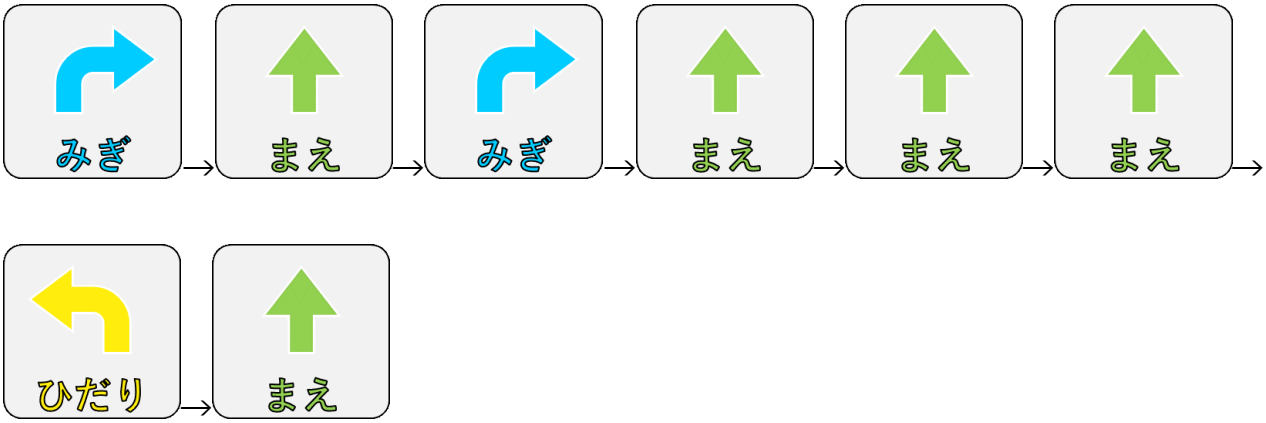
解説：

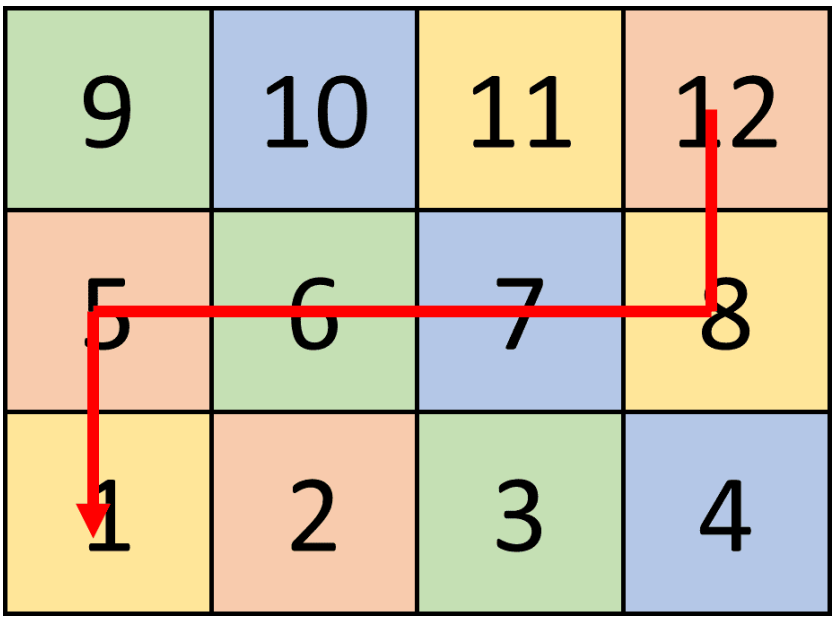
右折を繰り返すと左に行けることに気づくことがポイントです。自分ができることを組み合わせて目的を達成する方法を考えることは非常に重要です。他にも行き方はあるので探してみましょう。

3-3： 12 から 7 をとおって 1 へもどろう！

ただし、やじるしボタンは　9 かいまで　つかえるよ。

回答例：





解説：

矢印ボタンの使用回数に制限があるので、複数あるルートの中から条件を満たすルートを見つけなければいけません。

最初に見つけたルートがたまたま 9 回以内だとしてもそれで満足せず、それよりも少ない回数でゴールできるルートがないか探しましょう。いろいろな角度から物事を考えられる思考力が身につきます。

レベル：★★★★☆

出題意図：

レベル 4 はレベル 3 の目的を達成するときに分岐処理と反復処理を使用する問題です。反復処理をうまく使いこなせば繰り返し使用する命令を少ない回数でプログラミングできます。これまでの問題が少ない手間で解決できるようになると嬉しいですよ。

準備(じゅんび)：　ボットリー(ぼっとりー)を　1　に　うえ　むきで　おこう！

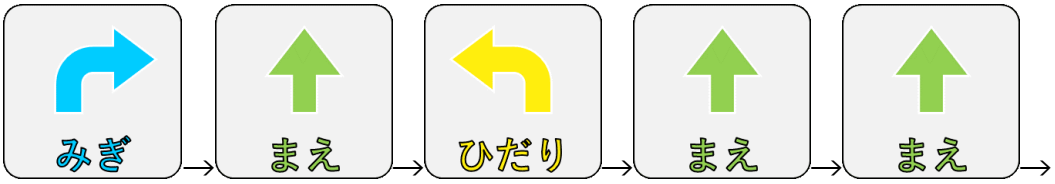
図(ず)の位置(いち)に　障害物(しょうがいぶつ)　をおこう！

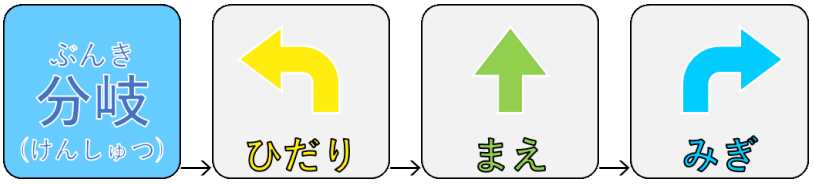
4-1：　1　から　6　をとおって　9　へいこう！

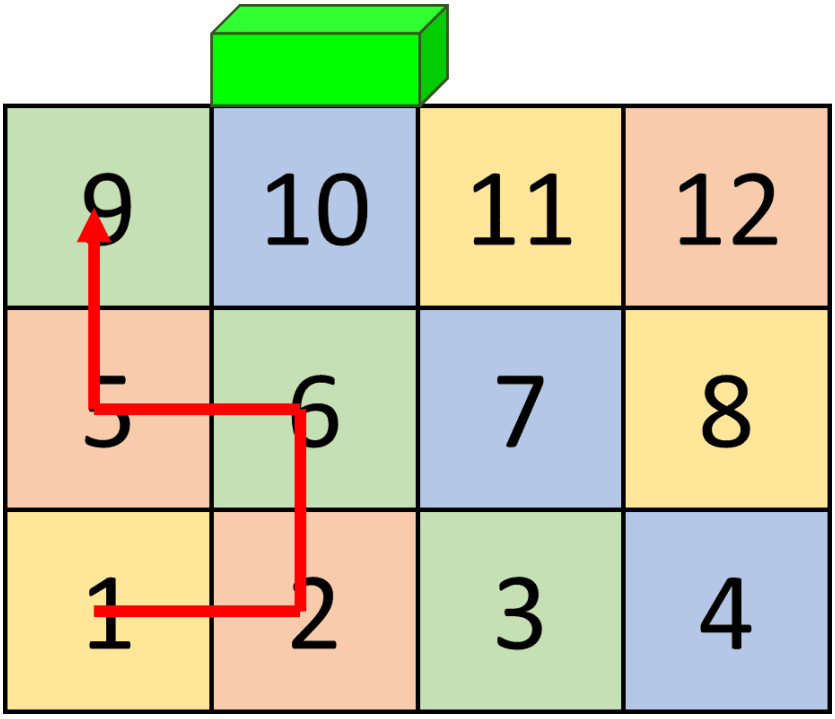
ただし、9　には　うえ　むきで　ゴール(ごーる)しよう！

さらに、障害物(しょうがいぶつ)　をよけるように　プログラミング(ぷろぐらみんぐ)してみよう！

回答例：





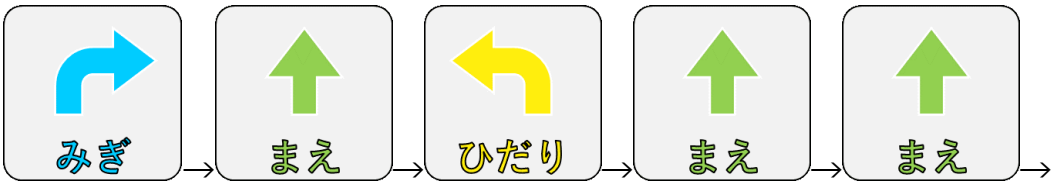


解説：

この問題は条件(障害物の有無)によって動作を変えることを学ぶ問題です。

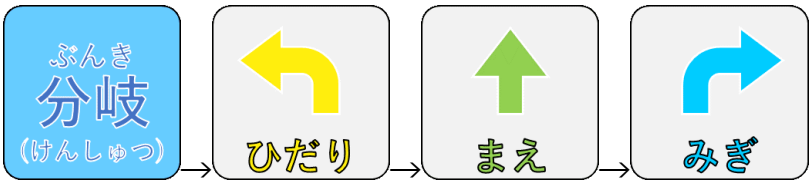
3-1 との違いは障害物を避けるようにプログラミングすることです。3-1 では 6 のマスに進入したら左折するようにプログラミングしていましたが、今回は障害物を検出したら左折するようにプログラミングします。

まずは、障害物が無いときの処理を入力します。



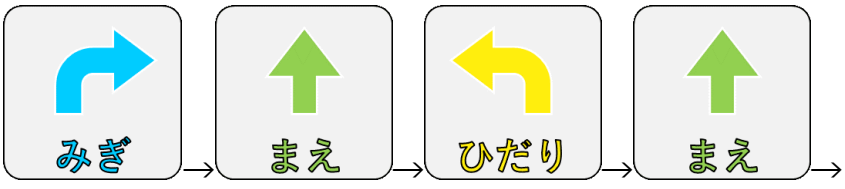
「右→前→左→前」が 6 のマスに進入するまで、次の「前」が障害物を検出して避けた後の行動です。実験したら、障害物を検出する距離は 1 マス半でしたので、ちょうど 6 のマスに進入したところで障害物を検出します。

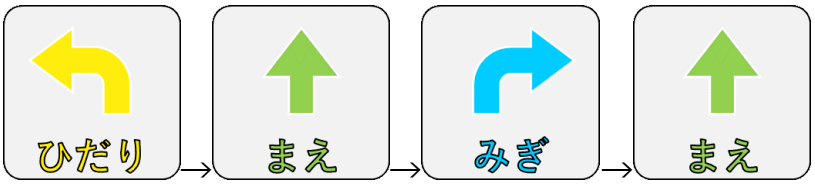
続けて障害物がある時の処理を入力します。



「分岐→左→前→右」は障害物を検出して、避けた後に前を向くまでです。

実際にプログラムを実行すると次の順番で命令が実行されます。



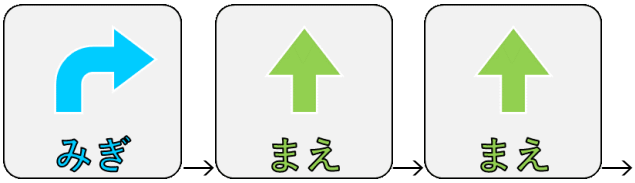


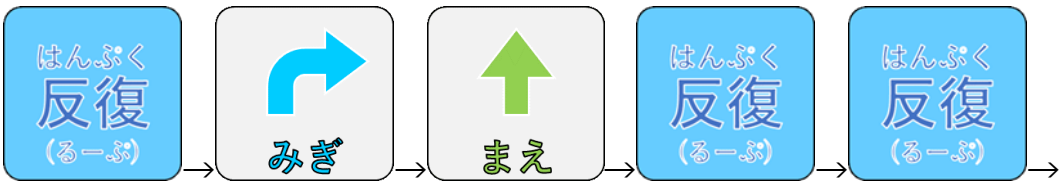
4-2： 9 から 7 をとおって 12 へいこう！

ただし、ひだり をつかわないで いこう！

さらに、るーぷ をつかってみよう！

回答例：

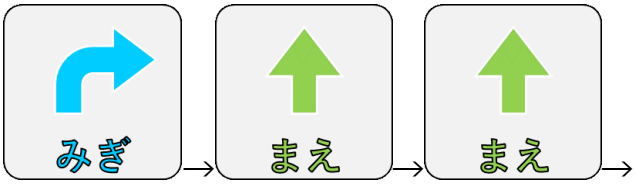


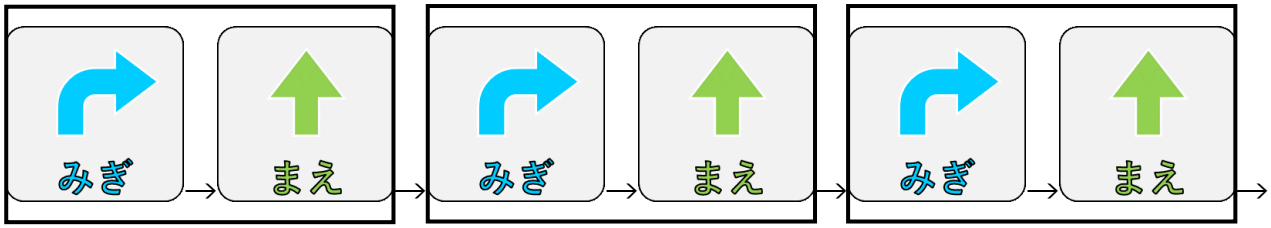


解説：

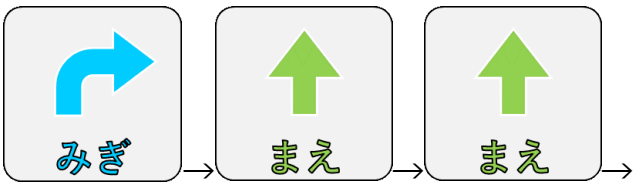
この問題は反復処理を使用すると少ない手間でプログラミングできることを学ぶ問題です。

3-2 の回答から連続で繰り返している箇所を見つけます。すると□で囲まれている部分が連続で繰り返していることがわかります。

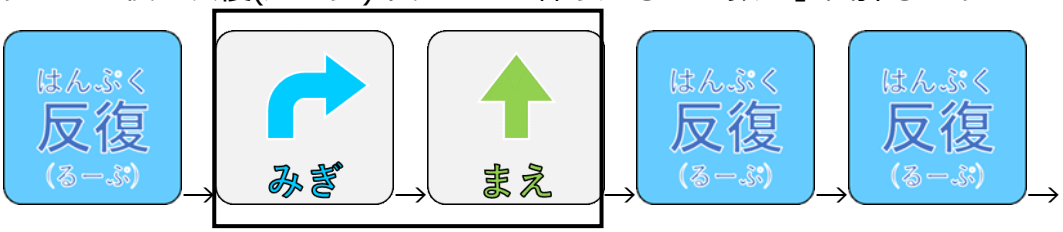




まずは、反復(ループ)を使わない場合と同じ様に入力します。



次に、反復させたい命令の前に反復(ループ)ボタンを入力し、反復させたい命令を入力します。その後に反復(ループ)ボタンを「繰り返したい数-1」回押します



反復(ループ)ボタンを押すと反復したい命令を入力できる状態になります。今回は「右→前」を入力します。3 回繰り返したいので、反復(ループ)ボタンを 2 回入力します。「繰り返したい数」ではなく「繰り返したい数-1」にするのは、既に 1 回分を入力しているからです。あと何回繰り返すか？と考えればわかりやすいです。

そして、最後の部分を続けて入力すれば完了です。

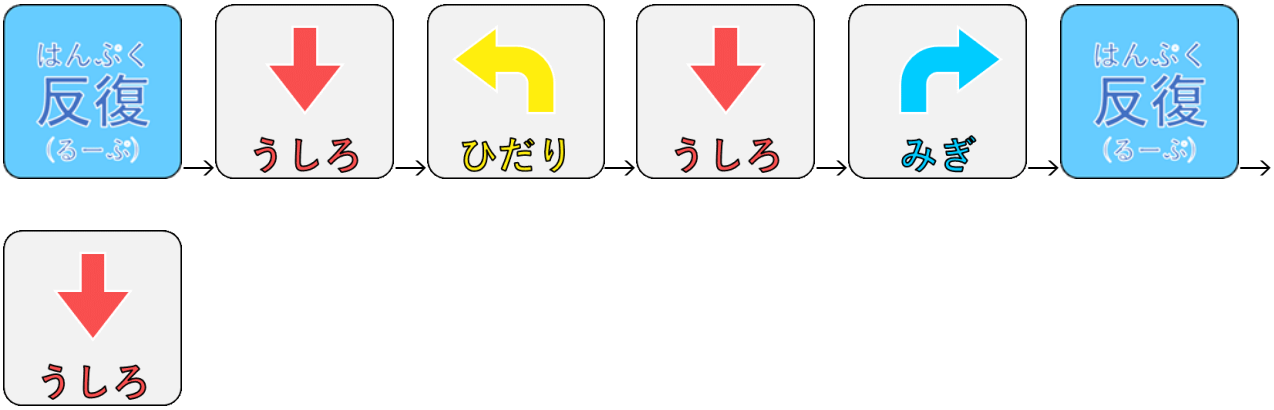
反復処理を使うことで少ない命令で目的を達成できました。今回は、繰り返したい命令が 2 個で、繰り返し回数が 3 回なので、命令回数はあまり減らせませんでしたが、繰り返したい命令と繰り返し回数多いと反復処理の便利さがわかります。

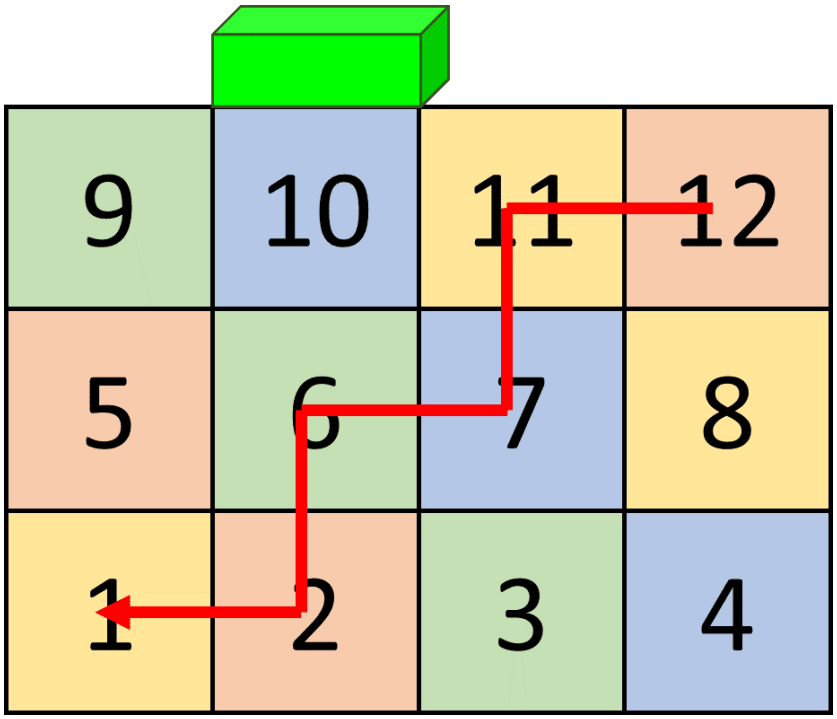
4-3：　12　から　7　をとおって　1　へもどろう！

ただし、やじるしボタンは　9かいまで　つかえるよ。

さらに、ループ　をつかってみよう！

回答例：



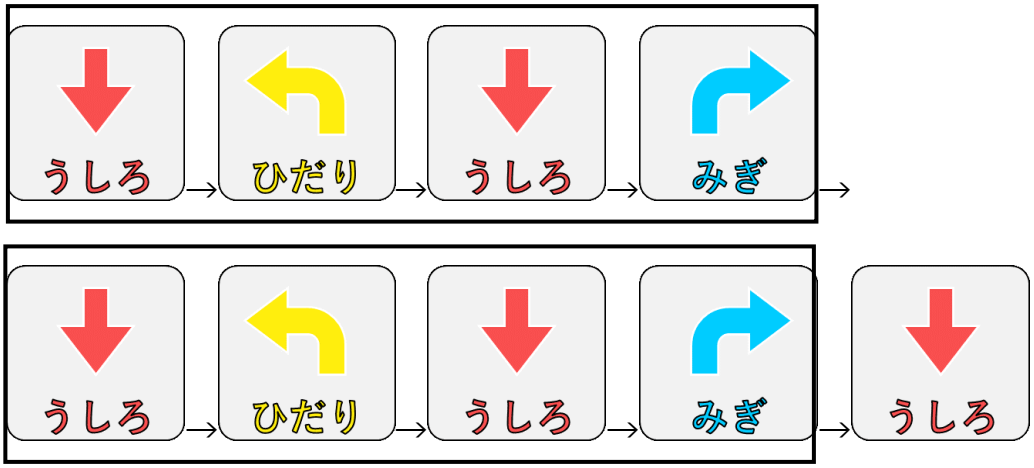


解説：

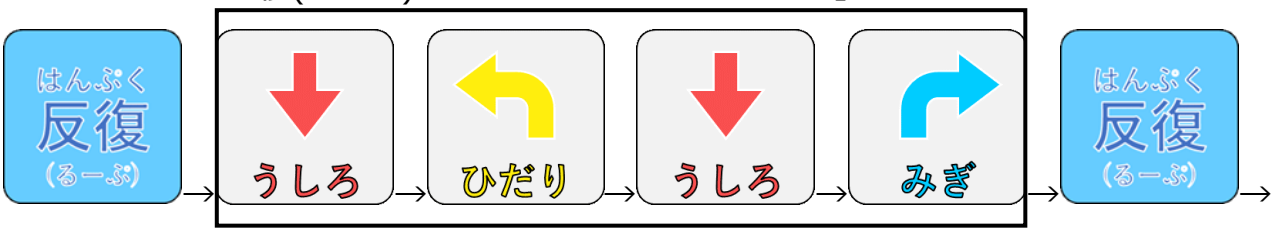
この問題は後ろ向きに進めば少ない命令でゴールできることに気づくことがポイントです。

3-3 の回答に反復(ループ)を使ってもいいのですが、1 命令しか減らす事ができません。なので、なるべく反復する部分が多くなるように色々なパターンを考えて導き出します。

今回導き出したパターンは次の通りです。その中から連続で繰り返している箇所を見つけます。すると□で囲まれている部分が連続で繰り返していることがわかります。



まずは、反復させたい命令の前に反復(ループ)ボタンを入力し、反復させたい命令を入力します。その後に反復(ループ)ボタンを「繰り返したい数-1」回押します



そして、最後の部分を続けて入力すれば完了です。

レベル：★★★★★

出題意図：

レベル 5 はこれまで学んだことをすべて活かして解決する問題です。現実世界では障害物の位置が予めわかっていることは少ないです。なので、障害物の位置や大きさが変わっても対応できるようなプログラムを予め作っておかなければなりません。レベル 5 は現実世界でも通用する考えを学ぶ問題です。

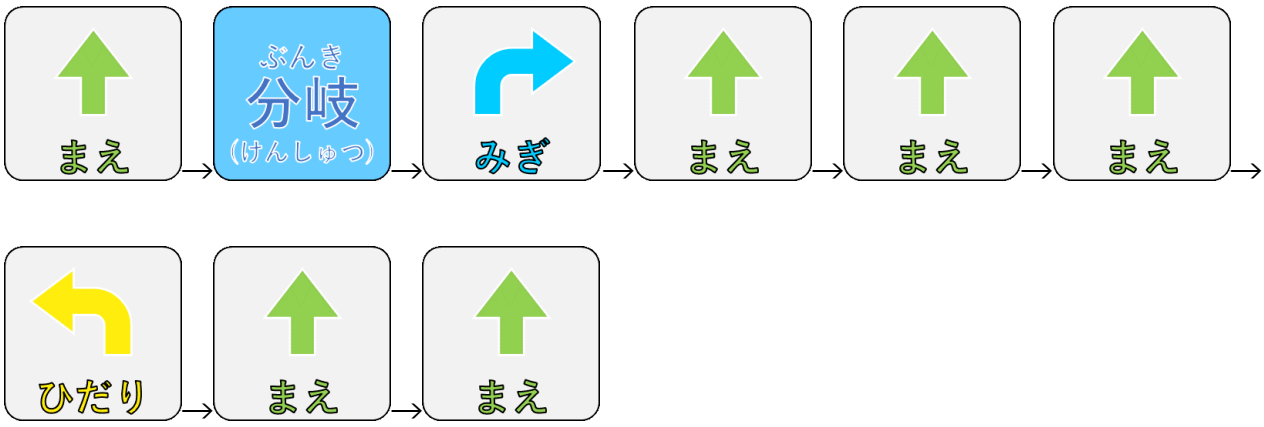
準備（じゅんび）：　ボットリー（ぼっとりー）を　スタート（すたーと）　に　うえ　むきで　おこう！

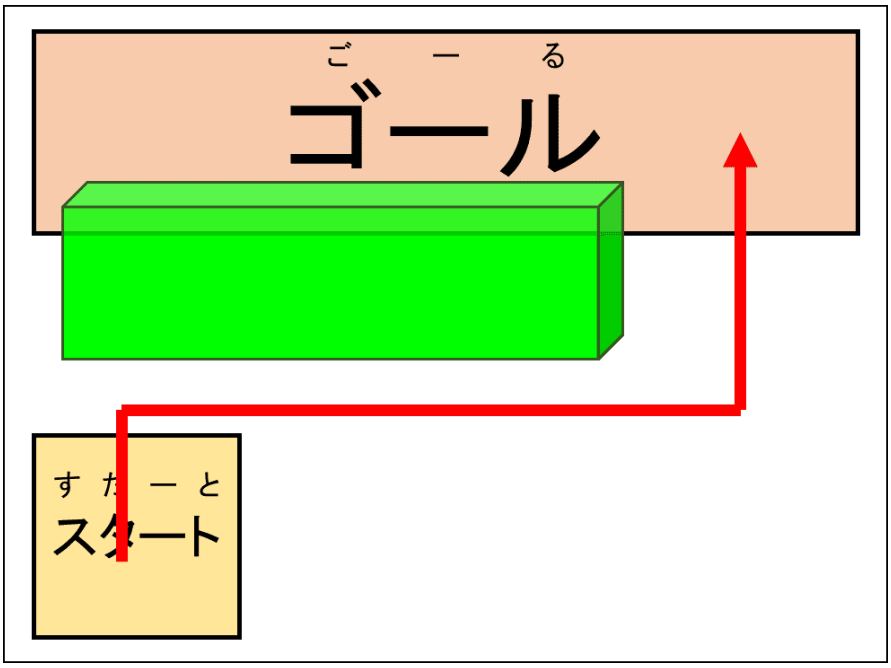
図（ず）のように　障害物（しょうがいぶつ）　をおこう！

5-1：　スタート（すたーと）　から　ゴール（ごーる）　へいこう！

ただし、さいしょは　まえ　にすすもう！

回答例：

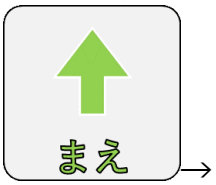




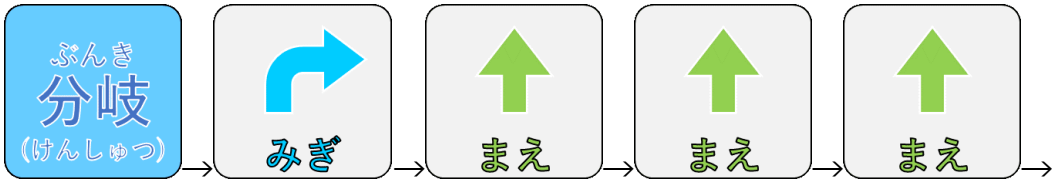
解説：

最初は前に進むことが指定されているため、前に進んでから障害物を検知し、障害物をよけるようにプログラミングします。

まずは、障害物が無いときの処理を入力します。



続けて障害物がある時の処理を入力します。障害物の大きさにあわせて右に進む距離を調整します。今回は 3 マス分です。3 マス移動したら、左を向いて前に進みゴールします。



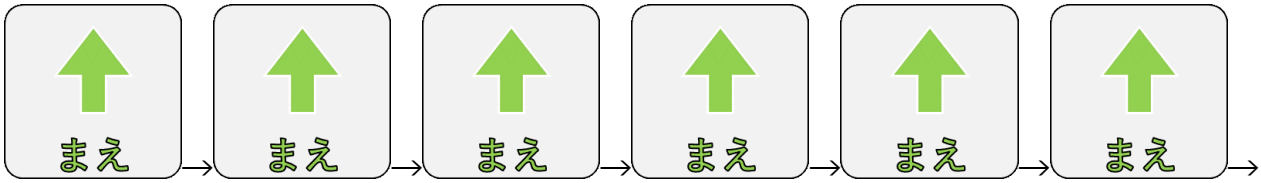
今回は障害物を検出してからゴールするまでをひとまとまりの処理にしましたが、障害物を検出したあとの処理を最小限にして、通常の処理を多くする方法もあるので、色々な方法を試してみてください。

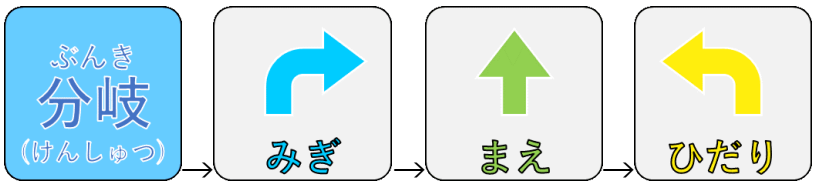
5-2： スタート から ゴール へいこう！

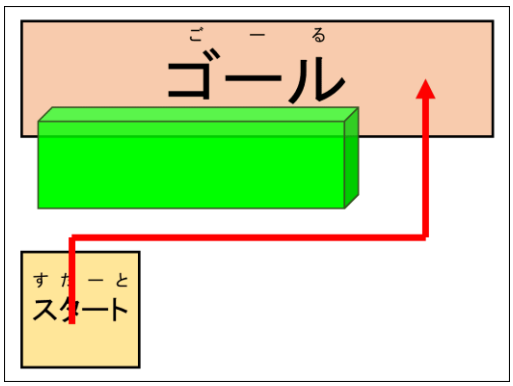
ただし、さいしょは まえ にすすもう！

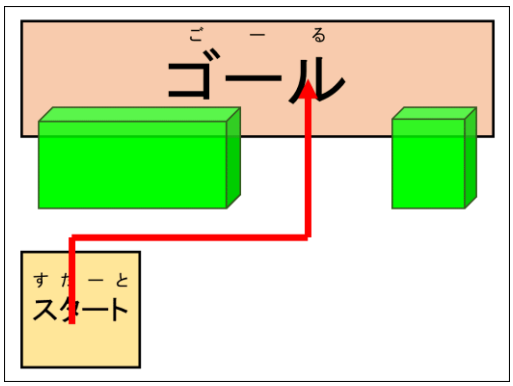
さらに、障害物 のおおきさが変わっても対応できるプログラムをつくろう！

回答例：









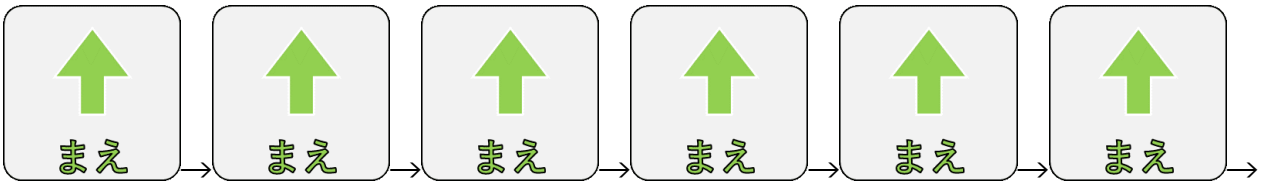
解説：

障害物の大きさが変わっても対応できるようにする必要があるため次のように考えます。

・前進して障害物があれば 1 マス分右に移動し、前進する。

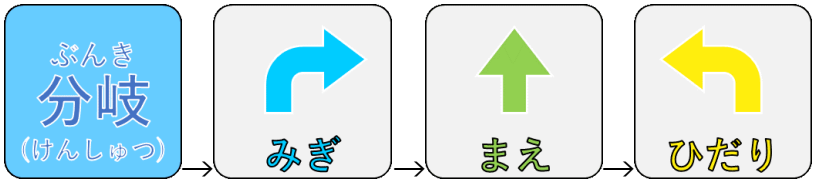
1 マス分右に移動し、前進することにチャレンジします。もし障害物があれば再び、1 マス分右に移動し、前進します。これを繰り返せば、障害物が無いところを前進することができます。

まずは、障害物が無いときの処理を入力します。



このおもちゃはゴールに着いたことを検出することができないため、多めに「前」を入れています。ゴールに着いたことを検出できるおもちゃであれば、「ゴールに着くまで前進する」といったプログラムを作ることができます。

続けて障害物がある時の処理を入力します。障害物の大きさはわからないので、1 マスずつ右に移動するようにします。



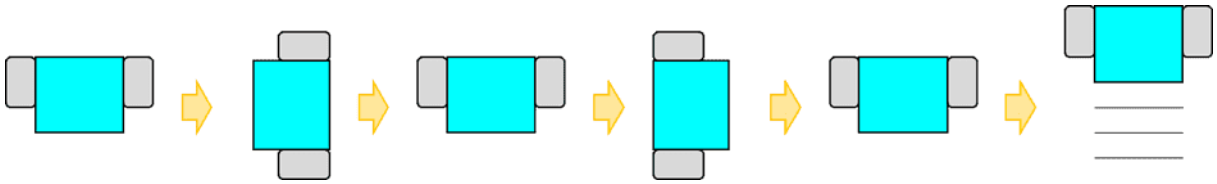
なので、プログラムを実行すると次の順番で命令が実行されます。

5-3：　スタート　から　ゴール　へいこう！

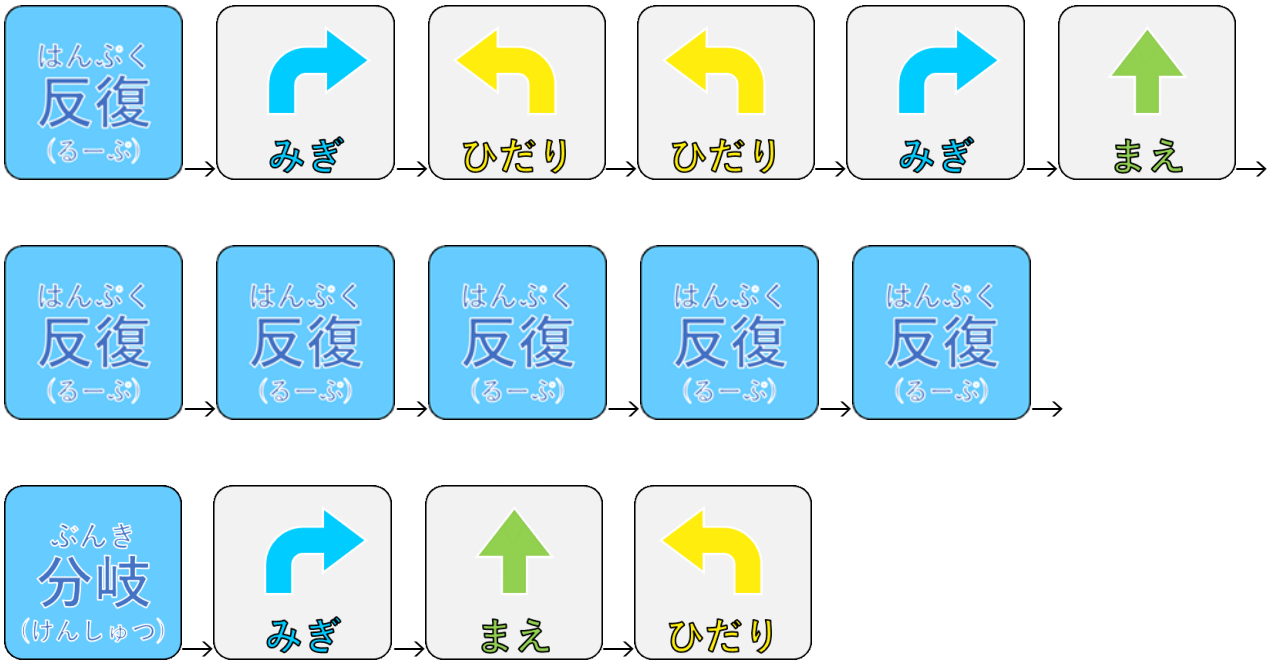
ただし、さいしょは　まえ　にすすもう！

さらに、障害物　のおおきさが変わっても対応できるプログラムをつくろう！

また、まえに　すすむまえは　みぎ　と　ひだり　をかくにんしよう！



回答例：

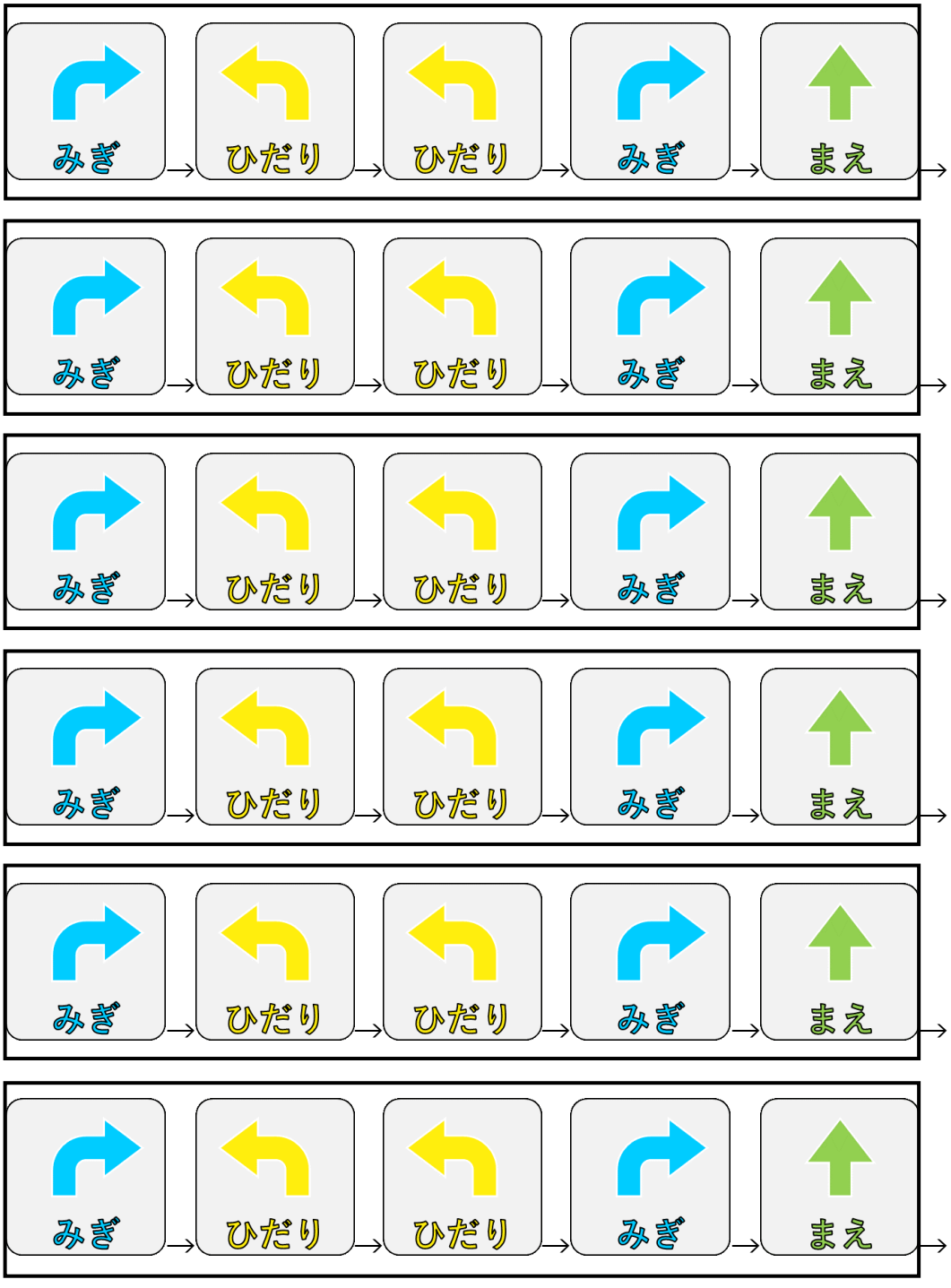


解説：

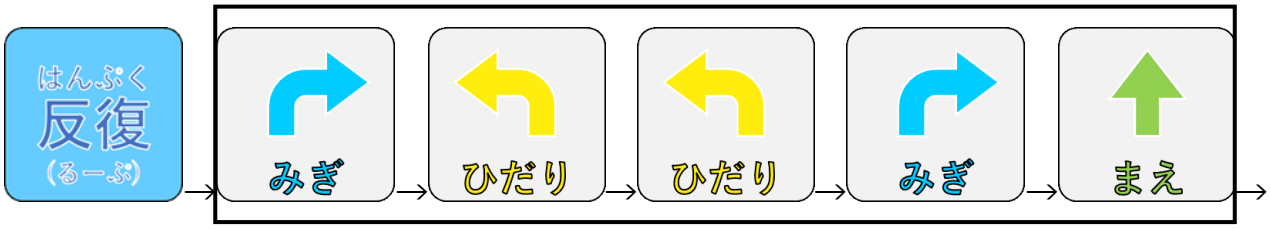
この問題は反復処理と分岐処理の両方を使う問題です。

いきなり反復処理を作るのが難しければ、まずは反復処理を使わずに命令を考えると考えやすいです。

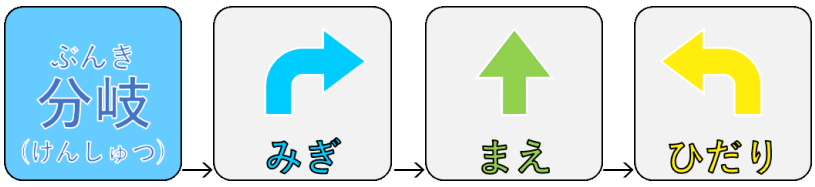
まずは 5-2 の障害物が無いときの処理に右と左を確認する動作を加えます。5-2 では前進を 6 回行っていたので、右と左を確認する動作を各前進処理の前に入れます。



次に、□で囲まれている部分を反復処理にします。



そして、障害物を検出したときの処理を行います。これは 5-2 と同じです。



これで完了です。分岐処理と反復処理を使う問題でした。分岐処理と反復処理を組み合わせたら複雑な動作もプログラミングすることができます。他にもいろいろな動作を作って楽しんでください。