【ロボ Basic2　07ー①　（90 分）】

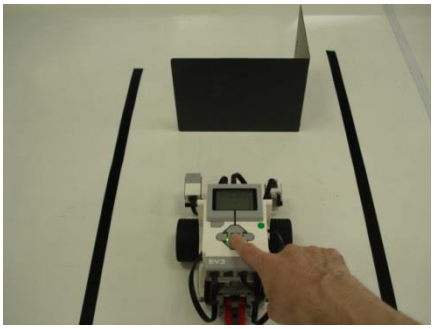
変数

はじめに

一部のコンピューターゲームでは、ゲームキャラクターの能力を数値で表すことがありますが、それと同じように、プログラミングでは様々なものの数値を保存しておき、その数値を使って計算したり、その数値自体を変更したりします。今日は、そのための数値の書き込みや読み込みの方法を学びましょう。

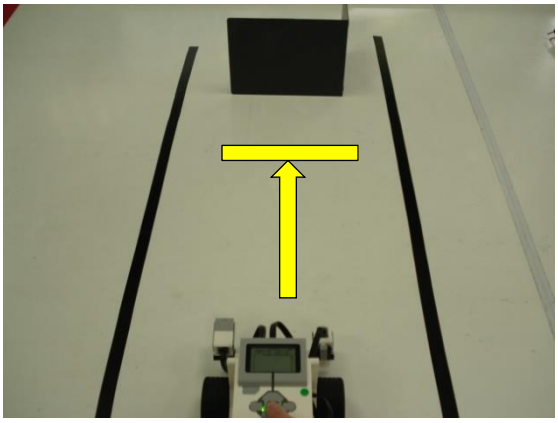
練習：先に覚えさせた位置のところまで前進する

カベの手前にロボットを置いてその位置をロボットに覚えさせ、少しはなれたところからその位置までロボットを前進させてみましょう。



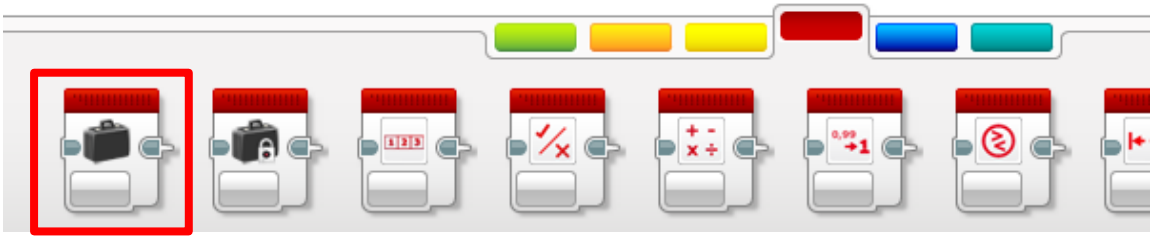
中央ボタンを押して位置を覚えさせる

ロボを置き直す

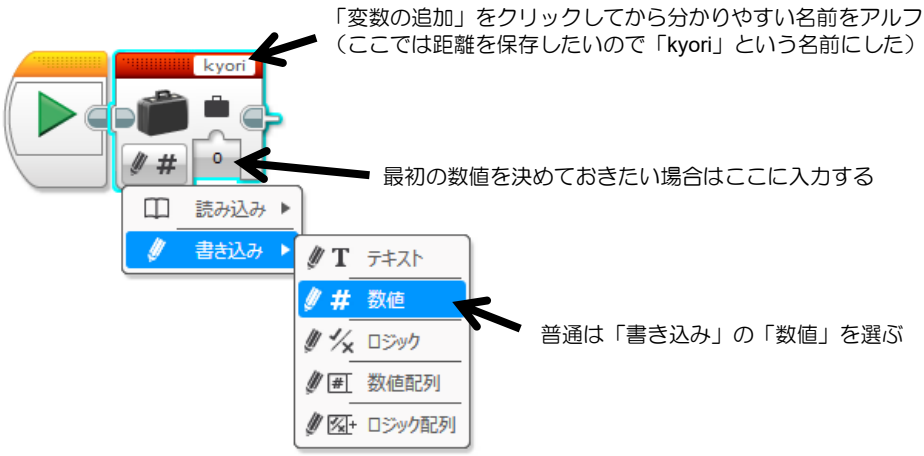


中央ボタンをもう一度押して
覚えさせた位置まで前進させる

位置を覚えさせるには、超音波センサーでカベまでの距離(きょり)を調べて、その数値を保存します。数値の保存には、赤グループの中の「変数」ブロックを使います。この変数が数をしまっておくためのカバンのような役割をします。



変数ブロックを使うには、それが何を保存するためのカバンなのかを先に決めておかなくてはいけません。まずは保存するものの種類とその名前を入れます。

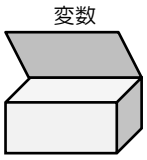


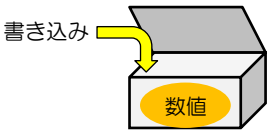
このようにして作ったカバン（変数）を使って数値を書き込んだり、読み込んだりします。

説明：変数

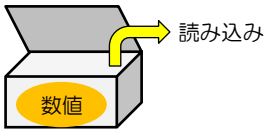
変数は数値（場合によっては文字なども）を１つ保存しておくための箱です。変数には必ずその数値が何の数値なのかを表すための名前をつけます。数値を変数に保存しておくと、いつでもその数値を変更（書きかえ）したり読み込んだりできます。

ゲーム（RPG）的に説明すると、「HP」「攻撃力」「防御力」などが変数の名前であり、その数値がその変数に保存されている数です。敵に攻撃するのであれば、「自分の攻撃力」から「敵の防御力」を引いてダメージを出して、そのダメージを「敵の HP」から引いて書き換えるといったような計算をしていきます。





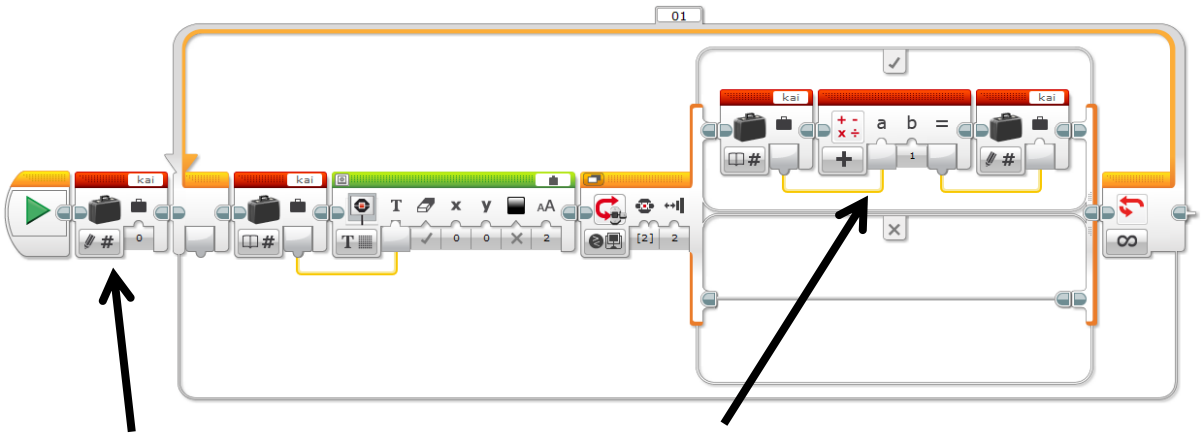
（数値がすでにはいっているときは、書きかえられます。）



（読み込みをしても、中の数値は消えません。）

練習：ボタンを押した回数を表示する

ロボットの中央ボタンを押した回数を画面に表示するプログラムを作ってみましょう。ループの中で変数を書き換えながらそれを表示するようにします。

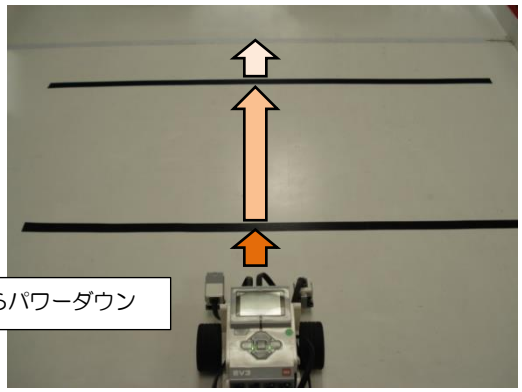


新しい名前の変数を作る
最初は 0 回なので「0」を入力する

ロボ中央のボタンを押すごとに変数の数が１だけ増える
押さなければ数はそのまま

練習：黒線をふむごとに遅くなる

ロボが黒線をふむ（超える）ごとに 10 ずつパワーダウンし、パワーが 0 以下になったら止まる（プログラム終了する）ようにプログラミングしてみましょう。



だいたい次のフローチャートのようにプログラムを作れば良いです。

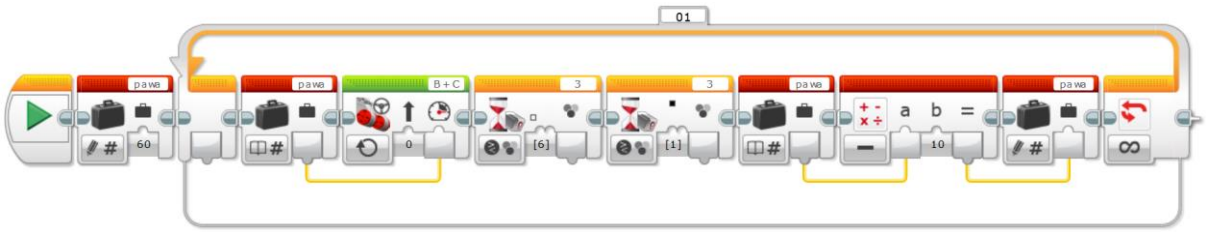
変数の準備（最初のパワーを入力）

→ 変数の数値をパワーにして前進オン → 白まで待機 → 黒まで待機
→ 変数の数を 10 減らす

変数の数が 0 以下になるまでループ

→ 前進オフ → 音（Game over）

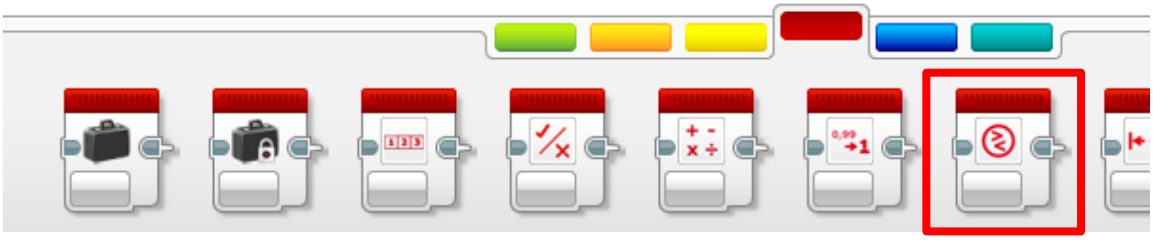
とりあえず、これまでに学んだことを思い出しながら、フローチャート通りに、ループ（とりあえず無限ループで良い）の中まで作りましょう。最初のパワーは、30～70 の間くらいで自由に設定してください。



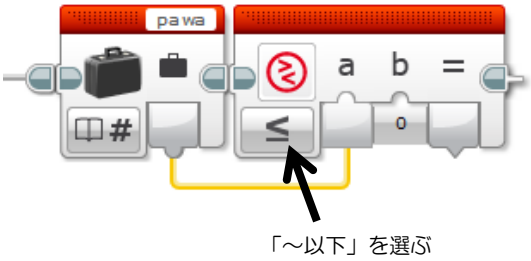
ここまで作れれば、とりあえず黒線をふむごとにパワーが下がってロボの動きが遅くなるようにはできます。しかし、このままでは無限ループであるため、パワーが 0 以下になってもゲームオーバーになりません。そこでパワーが 0 以下になったときにループが終わるようにします。

しかし、困ったことにこのプログラミングソフトでは、ループの終了の条件として選べるリストの中に「変数」がありません。

変数でループを終わらせるには、ループの最後のところで、赤グループの中の「比較」ブロックを使って、ループ終了の条件を別に作らなければいけません。

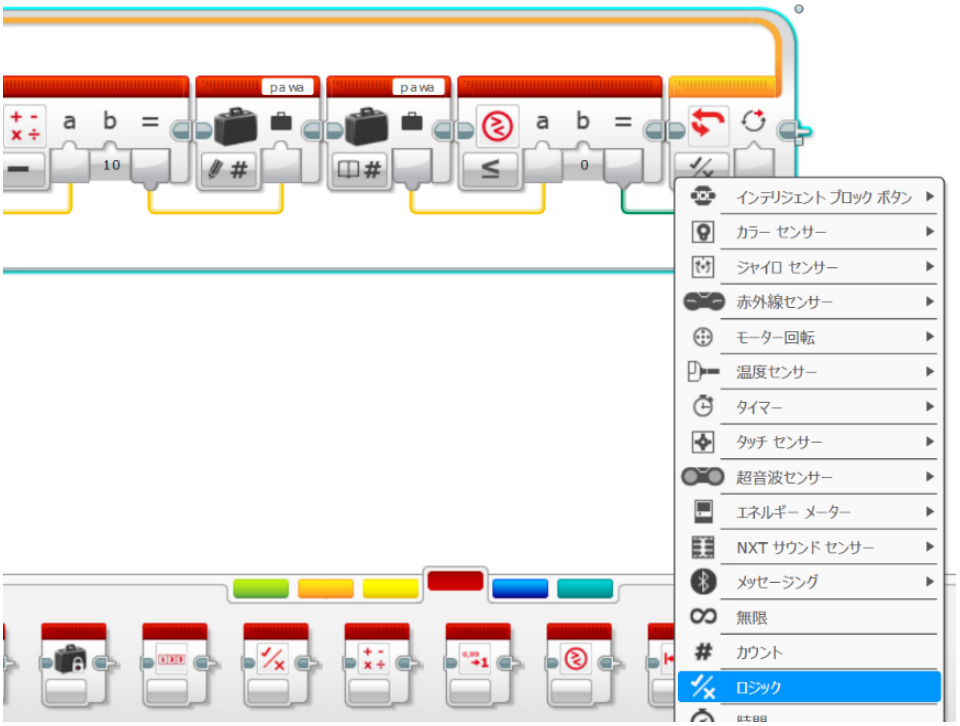


次のようにブロックを並べるとパワーの変数「pawa」が 0 以下かどうかを調べられます。

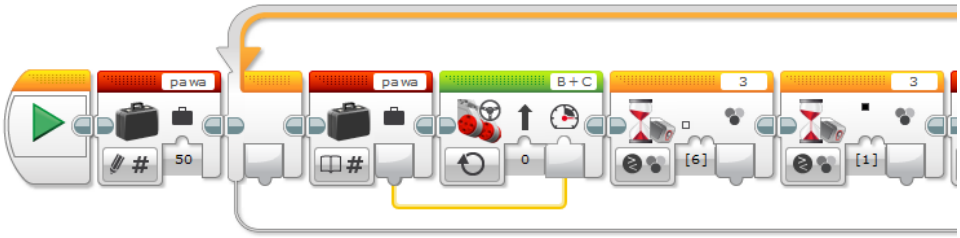


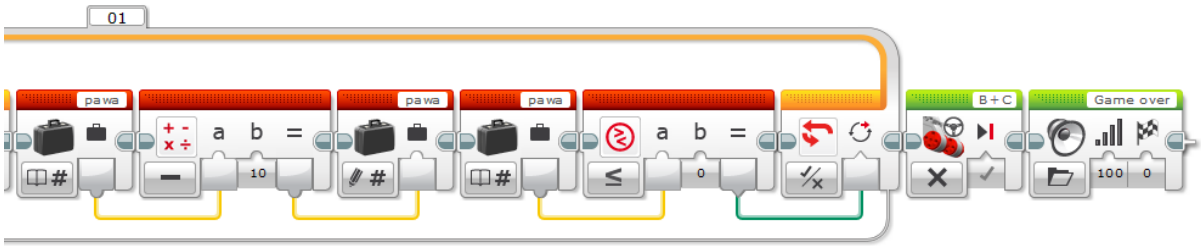
このように比較ブロックを使って「＝」のところに出てくる答えは、数値ではありません。答えは「正しい」（真：レ）か「まちがっている」（偽：×）かです。この答えをループの終了条件として使います。

ループの終了条件のリストからロジックを選び、そこに比較ブロックの答えを差し込むことで、その比較ブロックの答えが「正しい」（真）になったときにループを終わるようにできます。



ループを抜けられるようにできたら、あとはループの後ろに「ロボットの停止」と「ゲームオーバーの音」を加えるだけです。

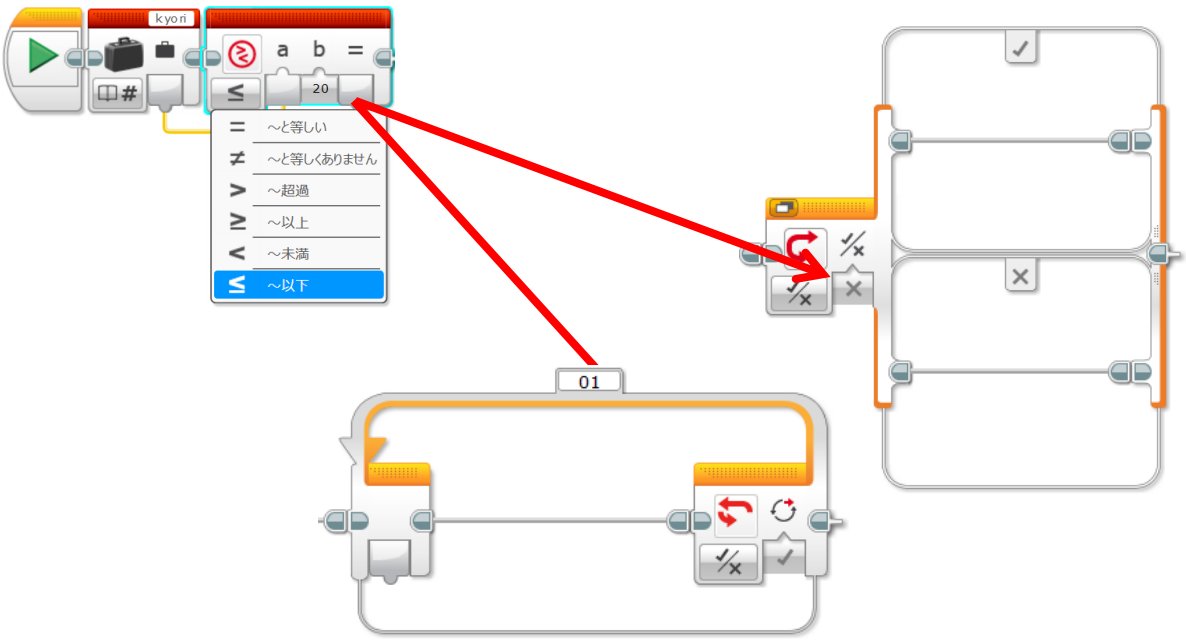




正しく動くことを確認したら、最初のパワーや黒線をふんだときに減るパワーの数値を変えて、何度か動かしてみましょう。

説明：比較（ひかく）ブロック

待機・ループ・スイッチのブロックの条件にとして、センサーで調べた数値を使いたいときは、それらのブロックの条件リストから対応するセンサーを選べば良いですが、変数の数値や計算で出した数値を条件とする場合は、同じようにできません。そのようなときに使うのが比較ブロックです。比較ブロックは、データワイヤーで他のブロックとつなぐことで条件を作ることができます。その条件で調べた結果を、スイッチやループの「ロジック」のマーク（「レ」や「×」）につなぎます。待機ブロックはリストに「ロジック」がないため、必要があればループブロックで待機ブロックと同等のものを作ります。

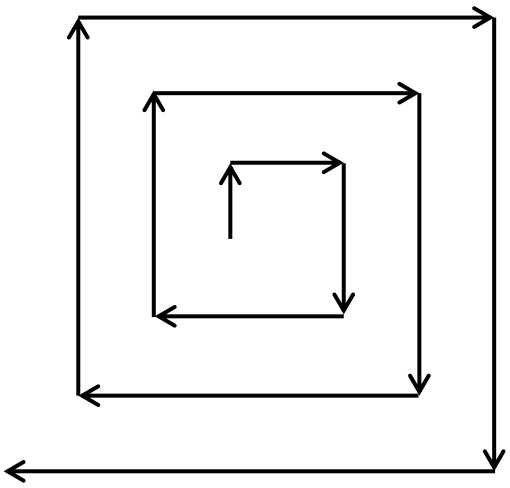


課題１：最初のカベまでの長さの２倍のところで止まる

カベの手前からロボをスタートさせ、最初のカベまでの長さをロボに覚えさせましょう。その後、覚えさせた長さの２倍の長さのところまで後退させて停止させましょう。（本体ボタンの押し込み待機をする必要はありません。）

課題２：だんだん大回りで四角に回る

右の図のようにだんだん大回りで四角にロボットが回るようにプログラミングしてみましょう。最初はタイヤ 0.5 回転分前進させ、そこから 0.5 回転分ずつ前進する長さが伸びる（0.5→1→1.5→2→…）ように作ってください。１つずつブロックを並べて作るのではなく、変数とループを使って効率良くプログラミングしましょう。



課題３：ボタンが押されるごとに加速してその場で回る

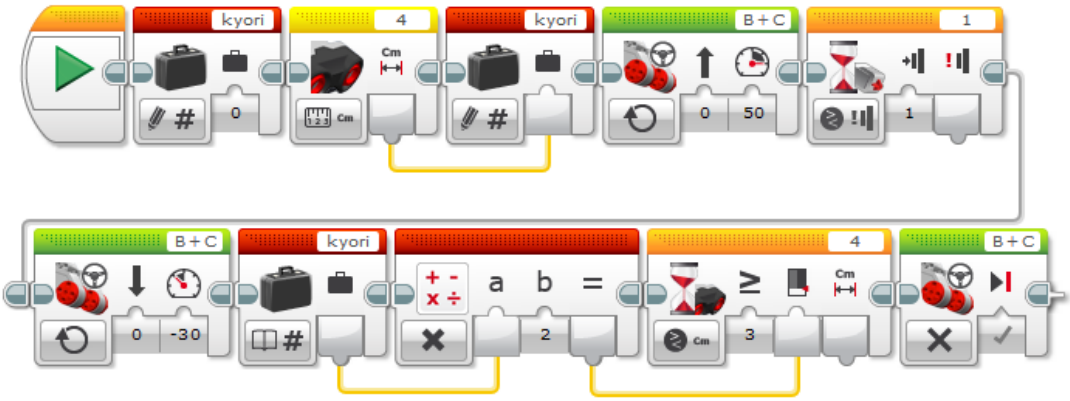
ロボットをステアリングカーブ 100 で動作（旋回）させ続け、本体上ボタンが押されるごとに、その動きが速くなるようにプログラミングしましょう。パワー10 からスタートさせ、ボタンが押されるごとにパワーが 10 ずつ増えるように作ってください。そして、パワーが 80 以上になったら停止して「Good-bye」と言うようにしてください（プログラム終了させてください）。

追加課題：ボタンが押されるごとに加速・減速してその場で回る

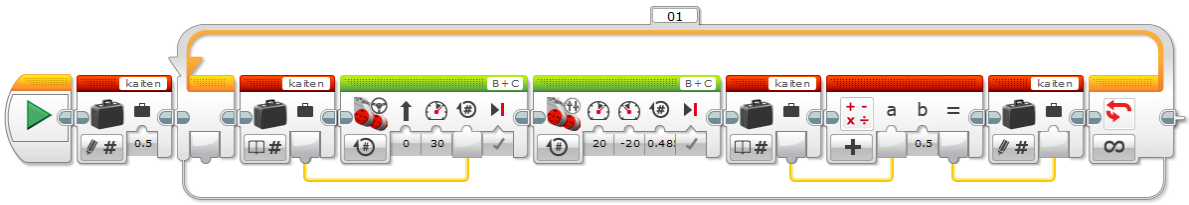
ロボットをステアリングカーブ 100 で動作（旋回）させ続け、本体上ボタンが押されるごとに動作のパワーが 8 ずつ増え、下ボタンが押されるごとに動作のパワーが 7 ずつ減るようにプログラミングしましょう。パワー40 からスタートさせ、動作中は本体の画面に現在のパワーを表示させるようにしてください。そして、パワーが 10～80 の範囲から出たら、停止して「Good-bye」と言うようにしてください（プログラム終了させてください）。

解答例（07－①）

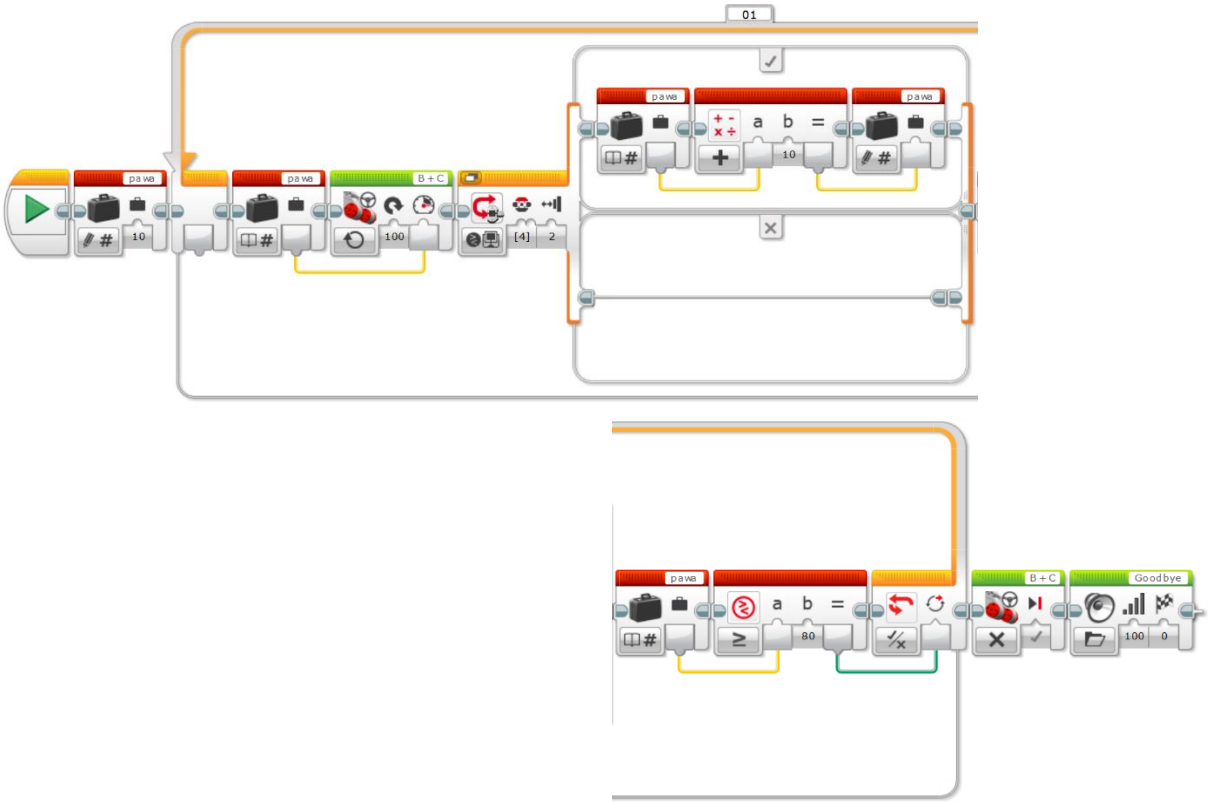
＜課題１＞



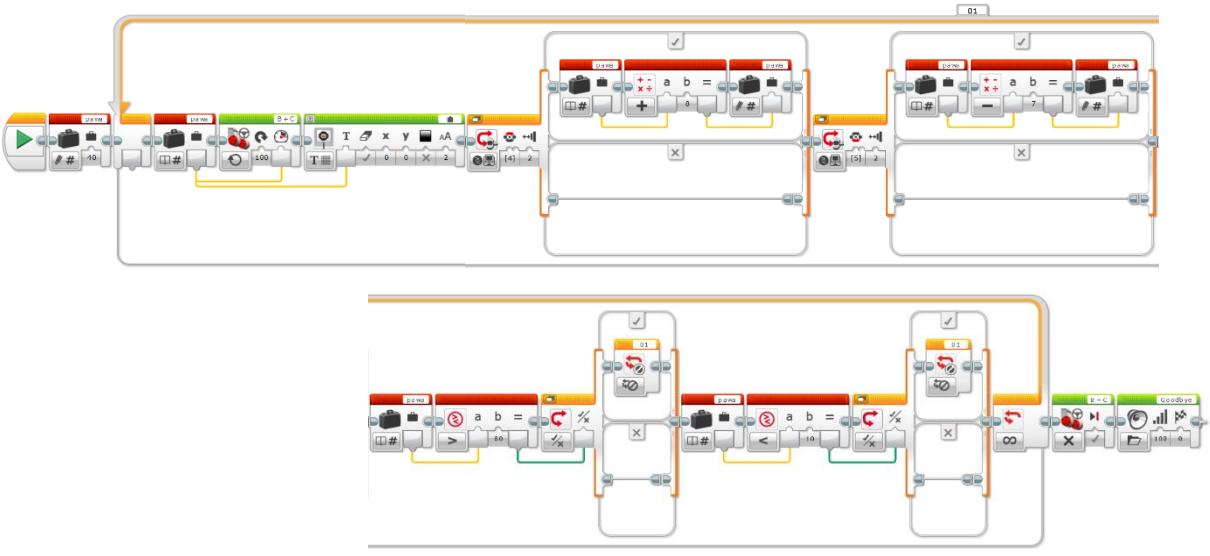
＜課題２＞



＜課題３＞


01
pawa
pawa
B + C
pawa
pawa
pawa
Goodbye

＜追加課題＞


01
pawa
B + C
pawa
pawa
pawa
pawa
pawa
pawa
pawa
B + C
Goodbye