情報関係基礎　第3問・第4問は，いずれか1問を選択し，解答しなさい。

第3問　（選択問題）　次の文章を読み，下の問い（問1～3）に答えよ。（配点　35）

吉野さんはロボットを操作して宝探しをするゲームを作成している。図1はゲーム画面の完成予想図である。ゲーム画面には，横YOKOマス×縦TATEマスで構成されたゲームボード（以降，ボードと呼ぶ），各種情報，ロボットを操作するためのボタンが表示される。ボードのマスには宝が一つ，罠が複数隠されている。ゲームの目的は，ボード上のロボットを上下左右に1歩（＝1マス）ずつ移動させ，罠を探知して避けながら，決められた操作回数以内で宝のマスに入れることである。

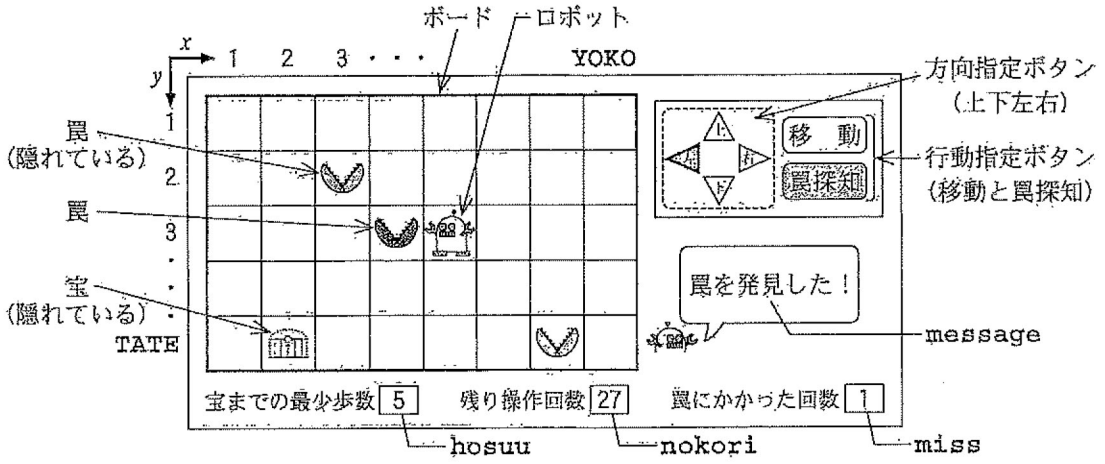

図1　ゲーム画面

学習指導要領（3）- 思・判・表 - イ
学習内容（3）- イ アルゴリズムとプログラム

問1　次の文章を読み，空欄［ア］～［カ］に入れるのに最も適当なものを，次ページの解答群のうちから一つずつ選べ。

ロボットと宝の位置は，マスの座標（x, y）で示す。ロボットの位置は変数robo_x, robo_y，宝の位置は変数takara_x, takara_yに格納する。ゲーム開始時にはロボットと宝は異なるマスに配置される。

ロボットの操作は，方向指定ボタンの一つを押し，続いて行動指定ボタンを押すことで行う。行動指定ボタンには移動ボタンと罠探知ボタンがある。図2は移動ボタンが押されたときの手続きである。方向指定ボタンで方向を指定して移動ボタンを押すと，指定した方向にロボットを移動させるための変数d_x, d_yに適切な値が代入されて図2が実行される。例えば，下方向を指定した場合は「d_x ← 0, d_y ← 1」，右方向を指定した場合は「d_x ← ［ア］, d_y ← ［イ］」の代入がされて図2が実行される。

変数nokoriには残り操作回数，変数messageにはゲームの状況を示す

メッセージを格納する。変数の値が更新されるとゲーム画面の数値やメッセージに反映される。図2では，ロボットのボード外への移動を防ぐ処理，残り操作回数を1減らす処理，宝のマスに入ったことを表示するための処理も行う。

```
(01)  もし robo_x + d_x > 0 かつ robo_x + d_x ≦ [ウ] かつ
          robo_y + d_y > 0 かつ robo_y + d_y ≦ [エ] ならば
(02)  |   robo_x ← robo_x + d_x
(03)  |   robo_y ← robo_y + d_y
(04)  を実行する
(05)  nokori ← nokori − 1
(06)  もし takara_x = robo_x かつ takara_y = robo_y ならば
(07)  |   message ←「宝を見つけた！　宝探し成功！」
(08)  を実行する
```

図2　移動ボタンが押されたときの手続き

また，ゲームのヒントとして，ロボットの位置から宝までの最少歩数を表示する。最少歩数は変数 hosuu に格納され，値が更新されるとゲーム画面の数値に反映される。図3は hosuu の計算手続きである。

```
(01)  sa_x ← takara_x − robo_x,  sa_y ← takara_y − robo_y
(02)  もし [オ] ならば sa_x ← sa_x × (-1) を実行する
(03)  もし [カ] ならば sa_y ← sa_y × (-1) を実行する
(04)  hosuu ← sa_x + sa_y
```

図3　宝までの最少歩数の計算手続き

[ア] ～ [カ] の解答群

⓪ 0	① 1	② -1	③ YOKO	④ TATE
⑤ sa_x < 0	⑥ sa_x > 0	⑦ sa_y < 0	⑧ sa_y > 0	
⑨ sa_x < YOKO		ⓐ sa_x > YOKO		
ⓑ sa_y < TATE		ⓒ sa_y > TATE		

情報関係基礎

学習指導要領（3）- 思・判・表 - イ
学習内容（3）- イ アルゴリズムとプログラム

問 2　次の文章を読み，空欄 キ ～ サ に入れるのに最も適当なものを，次ページのそれぞれの解答群のうちから一つずつ選べ。

ゲーム開始時にボードに隠されている罠は WANASUU 個である。ロボットが罠のマスに入ると罠にかかる。罠に 3 回かかると「宝探し失敗」となる。

罠の位置は，ロボットや宝と同じくマスの座標で表す。罠は複数個あるため，座標は配列で管理する。i(≧ 1)番目の罠の位置は，変数 Wana_x[i]，変数 Wana_y[i] で表す。なお，ゲーム開始時にはロボット，罠および宝はすべて異なるマスに配置される。

図 4 は，ロボットが罠のマスにいるかどうかを判定し，罠に 3 回かかったときに宝探し失敗のメッセージを表示するための手続きである。罠にかかった回数は，変数 miss で管理する。miss にはゲーム開始時に 0 を格納する。

```
(01)  i を 1 から [ キ ] まで 1 ずつ増やしながら，
(02)  |  もし Wana_x[i] = robo_x かつ
      |       Wana_y[i] = robo_y ならば
(03)  |  |  message ←「罠にかかった！　ダメージを受けた！」
(04)  |  |  [ ク ]
(05)  |  を実行する
(06)  を繰り返す
(07)  もし [ ケ ] = 3 ならば
(08)  |  message ←「3 回目だ！　ついに壊れた…宝探し失敗！」
(09)  を実行する
```

図 4　罠のマス判定手続き

罠は，初期状態では表示されていないが，プレイヤーは「罠探知」をすることで罠の有無を確認できる。図 2 に罠探知ボタンが押されたときの処理を追加して図 5 の手続きを作成した。図 5 では押された行動指定ボタンに応じて「移動」または「罠探知」を行っている。ここで追加された手続きにおいて，プレイヤーが方向指定ボタンの一つを押し，続いて罠探知ボタンを押した場合に，隣接する指定方向のマスに罠があるかどうかを調べ，罠があればその罠を表示状態に切り替える。なお，1 回の「罠探知」で，残り操作回数が 1 回減る。

ここでは罠の非表示と表示を管理する配列 Wana_hyoji を用意する。

Wana_hyoji[i]は，i番目の罠が非表示ならば0，表示ならば1とする。
Wana_hyoji の各要素は，ゲーム開始時に0で初期化する。

```
(01)　(指定した方向に対応する値をd_x，d_yに代入)
(02)　もし押されたボタンが移動ボタンならば
(03-10)　|(移動ボタンが押されたときの手続き(図2)と同じ)
(11)　を実行し，そうでなくもし押されたボタンが
　　罠探知ボタンならば
(12)　|  i を1から［キ］まで1ずつ増やしながら，
(13)　|  |  もし Wana_x[i] ＝ robo_x ＋ d_x かつ
　　|  |  |  Wana_y[i] ＝ robo_y ＋ d_y ならば
(14)　|  |  |  message ←「罠を発見した！」
(15)　|  |  |  ［コ］
(16)　|  |  を実行する
(17)　|  を繰り返す
(18)　|  ［サ］
(19)　を実行する
```

図5　行動指定ボタンが押されたときの「移動」と「罠探知」の手続き

［キ］，［ケ］の解答群

⓪ i	① robo_x	② robo_y
③ WANASUU	④ miss	

［ク］，［コ］・［サ］の解答群

⓪ miss ← 0	① miss ← 1	② nokori ← 0	③ nokori ← 1
④ miss ← miss ＋ 1		⑤ miss ← miss － 1	
⑥ nokori ← nokori ＋ 1		⑦ nokori ← nokori － 1	
⑧ Wana_hyoji[i] ← 0		⑨ Wana_hyoji[i] ← 1	
ⓐ Wana_hyoji[miss] ← 0		ⓑ Wana_hyoji[miss] ← 1	

情報関係基礎

学習指導要領（3）- 思・判・表 - イ
学習内容（3）- イ アルゴリズムとプログラム

問 3　次の文章を読み，空欄 シ ～ タ に入れるのに最も適当なものを，次ページのそれぞれの解答群のうちから一つずつ選べ。

図 6 は，これまでの手続きをまとめたゲーム全体の手続きである。ここで変数 zyotai はゲームの状態を表し，「プレイ中」なら 0，「宝探し成功」なら 1，「宝探し失敗」なら -1 が格納される。また，ゲーム開始時の残り操作回数は SYOKIKAISUU 回である。

```
(01)      zyotai ← 0, miss ← 0, nokori ← SYOKIKAISUU
(02)      message ←「 」
(03-06)   (宝までの最少歩数の計算手続き(図 3 )と同じ)
(07)      zyotai ＝ 0 の間,
(08)      |   【ロボットの行動指定待ち】
(09)      |   message ←「 」
(10-28)   |   (行動指定ボタンが押されたときの「移動」と「罠探知」の
          |    手続き(図 5 )と同じ)
(29-37)   |   (罠のマス判定手続き(図 4 )と同じ)
(38-41)   |   (宝までの最少歩数の計算手続き(図 3 )と同じ)
(42)      を繰り返す
```

図 6　宝探しゲームの手続き

吉野さんは，図 6 で宝を見つけたときに適切にゲームを終了させるために，zyotai ← 1 を シ に挿入し，罠に 3 回かかったときに適切にゲームを終了させるために，zyotai ← -1 を ス に挿入した。

次に，図 7 の残り操作回数判定の手続きを作成した。適切にゲームを終了させるために，図 7 の (03) 行目で セ を実行するようにした上で，図 6 の (29-37) 行目「罠のマス判定手続き」の直後に挿入した。その際，ソ という問題が生じた。そこで，その問題を解決するために，図 7 の (01) 行目の条件である「nokori ＝ 0」を「nokori ＝ 0 タ 」に修正した。

```
(01)  もし nokori ≠ 0 ならば
(02)  |   message ←「動きすぎて壊れた。宝探し失敗！」
(03)  |   セ
(04)  を実行する
```

図7　残り操作回数判定の手続き

シ ・ **ス** の解答群

⓪ 図2の(03)と(04)の間
① 図2の(07)と(08)の間
② 図2の(08)の下
③ 図4の(04)と(05)の間
④ 図4の(08)と(09)の間
⑤ 図4の(09)の下
⑥ 図5の(15)と(16)の間
⑦ 図5の(18)と(19)の間
⑧ 図5の(19)の下

セ の解答群

⓪ zyotai ← 0
① zyotai ← 1
② zyotai ← -1
③ miss ← 0
④ miss ← miss + 1
⑤ hosuu ← 0
⑥ hosuu ← hosuu − 1

ソ の解答群

⓪ 3回罠にかかったのに，「宝探し成功」になる場合がある
① 残り操作回数が0になったのに，ゲームが終了しない場合がある
② 宝のマスに入ったのに，「宝探し失敗」になる場合がある
③ 罠探知を1回すると，残り操作回数が2減る場合がある

タ の解答群

⓪ かつ zyotai ≠ 0
① かつ zyotai = 1
② かつ zyotai = -1
③ または zyotai = 0
④ または zyotai = 1
⑤ または zyotai = -1