1 【情報Ⅰ】 ベネッセ・駿台大学入学共通テスト模試と同傾向の出題

作成したプログラムを修正する問題という類似した形式の問題を出題

共通テスト　第３問　問３

```
(01)  Touchaku = [0, 3, 4, 10, 11, 12]
(02)  kyakusu = 要素数(Touchaku)
                                          ⓪
(03)  taiken を 1 から 15 まで 1 ずつ増やしながら繰り返す:
                                          ①
(04)      Kaishi = [0, 0, 0, 0, 0, 0]
(05)      Shuryou = [0, 0, 0, 0, 0, 0]
(06)      Shuryou[1] = taiken
(07)      i を 2 から kyakusu まで 1 ずつ増やしながら繰り返す:
(08)          Kaishi[i] = 最大値( カ , キ )
(09)          Shuryou[i] = ク
(10)      saichou = 0
(11)      i を 1 から kyakusu まで 1 ずつ増やしながら繰り返す:
(12)          saichou = 最大値( サ , ケ - コ )
(13)      もし シ ならば:
(14)          表示する("体験時間", taiken, "分間：",
                       "最長待ち時間", saichou, "分間")
                                          ②
                                          ③
```

図３　最長待ち時間が 10 分間未満となる体験時間を調べるプログラム

ある体験時間に対して最長待ち時間が 10 分間以上となった時点で，(04)行目以降の繰り返しをやめても実行結果は変わらない。そこでＹさんは，最長待ち時間が 10 分間以上となった時点で処理を止めるために，(03)行目を「(taiken <= 15) and (シ)の間繰り返す:」に変更した。「and」は「かつ」を意味する論理演算子であり，左右の式がともに真のときにだけ真となる。さらに，「taiken = 1」と「saichou = 0」を ス の位置に挿入した。また，「taiken = taiken + 1」を セ の位置に挿入した。

配列 Kaishi の初期化処理の実行回数は，修正前のプログラムでは 15 回だったが，修正後のプログラムでは ソ 回となった。

サ の解答群

⓪ Touchaku[i]	① Kaishi[i]
② Shuryou[i]	③ saichou

シ の解答群

⓪ saichou > 10	① taiken > 10
② saichou < 10	③ taiken < 10

第１回ベネッセ・駿台マーク模試　第３問　問３

Ｓ：さきほど作成したプログラムをいろんなパターンでテストしたのですが，選ぶマスによって数値と「▲」が一緒に表示されることがあります。

Ｔ：これはプログラム上の欠陥（バグ）ですね。プログラムを作成するときは，いろんなパターンでテストを行い，バグを発見・修正することは非常に大切です。今回このようになった原因は「+」の演算子には文字列を結合する役割もあるからです。仮に，変数 tensu に 80，変数 atai に「▲」が格納されている場合，「tensu = tensu + atai」の処理を行うと変数 tensu に格納される値は「80▲」となります。今回の図６のプログラムでは，シ にこの現象が起こってしまいます。

この現象を回避するためには，変数 tensu に数値のみが入るように図６の (15) 行目の条件を ス と修正するとよいでしょう。

シ の解答群

⓪ 今の点数が 100 点を超えているときに「▲」を選んだ場合
① 今の点数が 100 点以下のときに「▲」を選んだ場合
② 今の点数に関係なく「▲」を選んだ場合
③ 今の点数に関係なく「★」を選んだ場合

ス の解答群

⓪ そうでなくもし atai == "▲" ならば:
① そうでなくもし atai != "▲" ならば:
② そうでなくもし atai == "★" ならば:
③ そうでなくもし atai != "★" ならば:

両者はともに、問１・２で作成したプログラムについて、改良や不具合の修正のためにプログラムの修正を考えるという流れであった。（03）行目の「～1ずつ増やしながら繰り返す：」から「～の間繰り返す：」への修正は、一般的なプログラミング言語におけるfor文とwhile文に対応しており、この違いを理解できているかが解答するうえでのポイントであった。

2 【情報Ⅰ】 ベネッセ共通テスト対策教材と同傾向の出題

待ち行列のシミュレーションという問題設定を把握し，プログラムに落とし込む問題

共通テスト　第3問　問1・2

表1　昨年の来訪者の待ち時間を整理した結果

	到着時刻	開始時刻	終了時刻	待ち時間
1人目	0	0	3	0
2人目	3	3	6	0
3人目	4	6	ア	2
4人目	10	10	13	0
5人目	11	?	?	イ
6人目	12	?	?	ウ

（表の一部を“?”で隠してある）

（中略）

問2　次の文章を読み，空欄 カ ～ コ に入れるのに最も適当なものを，後の解答群のうちから一つずつ選べ。ただし，空欄 カ ・ キ ， コ は同じものを繰り返し選んでもよい。また，空欄 カ ・ キ の解答の順序は問わない。

次に，Yさんは問1で整理した考え方に基づき，昨年の6人目までの来訪者の到着時刻を用いて2人目以降の来訪者の待ち時間を求めるプログラムを作成した（図2）。

（中略）

```
(01)  taiken = 3
(02)  Touchaku = [0, 3, 4, 10, 11, 12]
(03)  kyakusu = 要素数(Touchaku)
(04)  Kaishi = [0, 0, 0, 0, 0, 0]
(05)  Shuryou = [0, 0, 0, 0, 0, 0]
(06)  Shuryou[1] = taiken
(07)  i を 2 から kyakusu まで 1 ずつ増やしながら繰り返す:
(08)  |  Kaishi[i] = 最大値( カ , キ )
(09)  |  Shuryou[i] = ク
(10)  |  表示する(i, "人目の待ち時間：",
      |              ケ - コ , "分間")
```

図2　2人目以降の来訪者の待ち時間を求めるプログラム

【出典】 2026年度大学入学共通テスト（本試験）より

2026直前演習　第5回第3問　問1・2

表1　さいころを投げてシミュレーションした結果（表は未完成である）

時刻（分）	さいころの目	的あて中	残り時間（分）	行列の人数（人）
0	1	①	4	0
1	4	①	3	0
2	2	①	2	1
3	2	①	1	2
4	4	②	4	1
5	2	②	3	ア
6	1	②	2	3

（中略）

```
(01)  shoyo_jikan = 4
(02)  nokori_jikan = 0
(03)  gyoretsu_ninzu = 0
(04)  max_ninzu = 0
(05)  エ を 0 から 180 まで 1 ずつ増やしながら繰り返す:
(06)  |もし 乱数(1,6)< オ ならば:
(07)  |  └ カ = カ +1
(08)  |もし nokori_jikan > 0 ならば:
(09)  |  └ nokori_jikan = キ
(10)  |もし nokori_jikan == 0 and gyoretsu_ninzu >= 1 ならば:
(11)  |  |nokori_jikan = ク
(12)  |  └ gyoretsu_ninzu = gyoretsu_ninzu-1
(13)  |もし ケ < gyoretsu_ninzu ならば:
(14)  └└ ケ = gyoretsu_ninzu
(15)  表示する (max_ninzu)
```

図1　行列が最も長くなったときの行列の人数を求めるプログラム

【出典】 2026直前演習情報Ⅰより

両者とも、シミュレーションとプログラミングの融合問題で、問1までで待ち行列の設定について表を埋めながら理解し、問2でそれをプログラムに落とし込む問題であった。変数の数が多いが問題文に丁寧な説明があるため、落ち着いて問題文を読み、アルゴリズムを正しく理解できるかがポイントであった。