問8　ビットの検査（データ構造及びアルゴリズム）　(H24春-FE 午後問8)

【解答】
［設問1］　a－ウ，b－ア
［設問2］　c－イ
［設問3］　d－ア，e－ア，f－オ

【解説】
8ビットのデータ中の指定したビット位置にあるビットの値を検査して結果を返すプログラム，及び8ビットのデータ中にある1のビットの個数を返すプログラムをテーマにした問題である。問題文中にある一つ一つのプログラムは短いものなので，プログラムの説明をよく読んで，ビットの論理演算やシフトなどの知識を活用しながら，ビット列の変化をトレースすれば解答できる内容である。また，設問3の処理量の問題については，繰返しが最小になる場合と最大になる場合について考えれば解答できる。問題の難易度はやや易しいといえる。

［設問1］
最初に，問題文の例を使ってビット検査の方法を確認する。

(例1)									(例2)								
ビット番号	7	6	5	4	3	2	1	0	ビット番号	7	6	5	4	3	2	1	0
Data	0	1	0	1	0	1	0	1	Data	0	0	1	1	0	0	1	1
Mask	1	1	1	0	0	0	0	0	Mask	0	0	0	1	0	0	0	1
返却値	1								返却値	2							

（例1）では，Maskのビット番号7～5の3ビットが1なので，Dataのこの3ビットの値を検査する。そして，検査対象のビットには0と1が混在しているので，返却値は1となる。また，（例2）では，Maskのビット番号4と0が1なので，Dataのビット番号4と0の値を検査するが，どちらのビットも1なので返却値は2となる。
解答群の内容を見ると，この問題では，Data（検査データ）とMask（検査するビット位置が1，他のビットが0：マスクビット）の論理積（AND）か論理和（OR）を求めて，対象となるビットの値を検査しているようである。どちらの演算なのかを考えるために，（例1），（例2）のDataとMaskの論理積，論理和を求めると次のようになる。

(例1)									(例2)								
ビット番号	7	6	5	4	3	2	1	0	ビット番号	7	6	5	4	3	2	1	0
Data	0	1	0	1	0	1	0	1	Data	0	0	1	1	0	0	1	1
Mask	1	1	1	0	0	0	0	0	Mask	0	0	0	1	0	0	0	1
論理積	0	1	0	0	0	0	0	0	論理積	0	0	0	1	0	0	0	1
論理和	1	1	1	1	0	1	0	1	論理和	0	0	1	1	0	0	1	1

この結果から分かるように，論理積の場合は，Data中のビット列から，Maskの値が1になっているビット位置の値だけがそのまま取り出され，その他のビットは全て0となる。一方，論理和の場合は，Maskの値が0になっているビット位置の値だけがそのまま取り出され，その他のビットは全て1となってしまう。このことから，あるビット列から特定のビット位置の値だけを取り出すには，検査対象のビット列と対応するビットに1，それ以外のビットに0を設定したマスクビットとの論理積（AND）を求めればよいことが分かる。

［プログラム1］
```
○整数型関数:BitTest(8ビット論理型:Data, 8ビット論理型:Mask)
○整数型:RC          /* 返却値 */
  ▲ a
  ・RC ← 2          /* 返却値は2 */
  ▲ b
  ・RC ← 0          /* 返却値は0 */
  ・RC ← 1          /* 返却値は1 */
・return RC         /* RCを返却値として返す */
```

・空欄a：空欄aが真の場合は返却値に2が設定されているので，空欄aではData中の検査したビットが全て1であることを判断できる条件を考える。前述の説明にもあるように，検査するビットの値だけを取り出すためには，検査対象のデータ（Data）とマスクビット（Mask）との論理積を求める。その結果，検査したビットだけが1，それ以外のビットは全て0であったときに，検査したビットが全て1であったと判断できる。そして，それは結果がMaskと同じ値ということである。したがって，条件式は(Data & Mask) = Maskと表現でき，（ウ）が正解である。
・空欄b：空欄bが真の場合は返却値に0（検査した全てのビットが0），偽の場合は返却値に1（検査したビット中に0と1が混在）を設定している。前述の説明にもあるように，Mask と Data の論理積を求めると，検査対象のビットはそのままの値，対象外のビットは全て0になるので，検査対象のビットが全て0の場合の演算結果は"00000000"Bに，1が含まれる場合には，それ以外の値になる。したがって，空欄bの条件式は(Data & Mask) ="00000000"Bと表現でき，（ア）が正解である。

［設問2］
プログラム1では，Mask中に1のビットが1個以上あることを前提としていたが，この前提を取り除いて，Mask中の1のビットが0個の場合は，返却値0を返すようにプログラムを変更する。この変更のための修正案は①～③の三つあり，適切なものを選択する。なお，修正案中の空欄a，bは，それぞれ設問1で解答した“(Data & Mask) = Mask”，“(Data & Mask) = "00000000"B”が入ることに注意する。
Mask中の1のビットが0個の場合にDataとMaskの論理積を求めると，演算結果のビット列は全て0になるため，その判定条件は(Data & Mask) = "00000000"Bと表現できる。また，Mask自体のビット列が"00000000"Bであるため，(Data & Mask) = Maskとも表現できる。このことを基に修正案を順番に検証していく。

修正案①（変更なし）
```
▲(Data & Mask) = Mask
・RC ← 2
  ▲(Data & Mask) = "00000000"B
  ・RC ← 0
  ・RC ← 1
・return RC
```

修正案①では，Maskのビット列が全て0の場合，空欄a（(Data & Mask) = Mask）が真となるので，本来は返却値として0を返すべきところを2を返すことになり，正しい結果とはならない。

修正案②
```
▲(Data & Mask) = "00000000"B
・RC ← 0
  ▲(Data & Mask) = Mask
  ・RC ← 2
  ・RC ← 1
・return RC
```

修正案②では，最初の判定条件が“(Data & Mask) = "00000000"B”なので，Maskのビット列が全て0の場合には返却値が0となる。また，検査した全てのビットが0の場合もこの条件が成り立つので，返却値0に関する判定は正しく行うことができる。そして，この条件が偽の場合には，続いて“(Data & Mask) = Mask”の判定を行うが，その結果は，検査したビットが全て1の場合はMaskと同じビット列に，0と1が混在する場合はMaskと異なるビット列になるため，真のときに2，偽のときに1と正しい返却値を返している。したがって，正しく動作する。

修正案③
```
・RC ← 1
▲(Data & Mask) = "00000000"B
・RC ← 0
▲(Data & Mask) = Mask
・RC ← 2
・return RC
```

修正案③では，最初の判定条件である“(Data & Mask) = "00000000"B”が真となった場合にも，続けて“(Data & Mask) = Mask”の判定を行うことになる。修正案①の説明にもあるように，Mask中の1のビットが0個の場合にはこの二つの判定条件はどちらも真となるため，結果として返却値が2となってしまい，正しい結果とはならない。
以上の内容から，正しく動作するのは修正案②だけであることが分かり，空欄cは（イ）が正解である。

［設問3］
プログラム2，3の処理効率について考える。

［プログラム2］
```
（処理量）
        ○整数型関数:BitCount (8ビット論理型:Data)
        ○8ビット論理型：Work
        ○整数型:Count, Loop
①  1    ・Work ← Data
②  1    ・Count ← 0
③  4    Loop: 0, Loop < 8, 1
④  3      ▲Workの最下位ビットが1
⑤  1      ・Count ← Count + 1
⑥
⑦  1    ・Workを右へ1ビット論理シフトする
⑧
⑨  2    ・return Count          /* Countを返却値として返す */
```

※○付き数字は行番号を表す。

・空欄d：整数型関数BitCountは，8ビットのデータ中にある1のビットの個数を返す。プログラム2は，最下位ビットが1かどうかを判断することで1のビットの個数をカウントするものであるが，ビットの判定の都度，ビット列を1ビットずつ右へシフトすることで，ビット番号2，ビット番号3，…，ビット番号7を順番に最下位ビットへ移動させ，8ビット全てについての判定を行う。そのため，行番号③の繰返しの判断は9回（Loopは0から始まりLoop=8で繰返し処理を抜ける），行番号④，⑦の処理は8回実行される。また，行番号⑤の処理はWorkの最下位ビットが1の場合だけ実行されるので，実行回数が最小になるのはWorkのビットが全て0の場合，逆に最大になるのはWorkのビットが全て1の場合であり，実行回数はそれぞれ0回，8回となる。これを整理すると，最小，最大の処理量は次のようになる。

行番号	処理量	最小の処理量	最大の処理量
①	1	1×1回=1	1×1回=1
②	1	1×1回=1	1×1回=1
③	4	4×9回=36	4×9回=36
④	3	3×8回=24	3×8回=24
⑤	1	1×0回=0	1×8回=8
⑦	1	1×8回=8	1×8回=8
⑨	2	2×1回=2	2×1回=2
		合計　72	合計　80

問8　ビットの検査（データ構造及びアルゴリズム）　(H24春-FE 午後問8)

【解答】
［設問1］　a－ウ，b－ア
［設問2］　c－イ
［設問3］　d－ア，e－ア，f－オ

【解説】
8ビットのデータ中の指定したビット位置にあるビットの値を検査して結果を返すプログラム，及び8ビットのデータ中にある1のビットの個数を返すプログラムをテーマにした問題である。問題文中にある一つ一つのプログラムは短いものなので，プログラムの説明をよく読んで，ビットの論理演算やシフトなどの知識を活用しながら，ビット列の変化をトレースすれば解答できる内容である。また，設問3の処理量の問題については，繰返しが最小になる場合と最大になる場合について考えれば解答できる。問題の難易度はやや易しいといえる。

［設問1］
最初に，問題文の例を使ってビット検査の方法を確認する。

(例1)									(例2)								
ビット番号	7	6	5	4	3	2	1	0	ビット番号	7	6	5	4	3	2	1	0
Data	0	1	0	1	0	1	0	1	Data	0	0	1	1	0	0	1	1
Mask	1	1	1	0	0	0	0	0	Mask	0	0	0	1	0	0	0	1
返却値	1								返却値	2							

（例1）では，Maskのビット番号7～5の3ビットが1なので，Dataのこの3ビットの値を検査する。そして，検査対象のビットには0と1が混在しているので，返却値は1となる。また，（例2）では，Maskのビット番号4と0が1なので，Dataのビット番号4と0の値を検査するが，どちらのビットも1なので返却値は2となる。
解答群の内容を見ると，この問題では，Data（検査データ）とMask（検査するビット位置が1，他のビットが0：マスクビット）の論理積（AND）か論理和（OR）を求めて，対象となるビットの値を検査しているようである。どちらの演算なのかを考えるために，（例1），（例2）のDataとMaskの論理積，論理和を求めると次のようになる。

(例1)									(例2)								
ビット番号	7	6	5	4	3	2	1	0	ビット番号	7	6	5	4	3	2	1	0
Data	0	1	0	1	0	1	0	1	Data	0	0	1	1	0	0	1	1
Mask	1	1	1	0	0	0	0	0	Mask	0	0	0	1	0	0	0	1
論理積	0	1	0	0	0	0	0	0	論理積	0	0	0	1	0	0	0	1
論理和	1	1	1	1	0	1	0	1	論理和	0	0	1	1	0	0	1	1

この結果から分かるように，論理積の場合は，Data中のビット列から，Maskの値が1になっているビット位置の値だけがそのまま取り出され，その他のビットは全て0となる。一方，論理和の場合は，Maskの値が0になっているビット位置の値だけがそのまま取り出され，その他のビットは全て1となってしまう。このことから，あるビット列から特定のビット位置の値だけを取り出すには，検査対象のビット列と対応するビットに1，それ以外のビットに0を設定したマスクビットとの論理積（AND）を求めればよいことが分かる。

［プログラム1］
```
○整数型関数:BitTest(8ビット論理型:Data, 8ビット論理型:Mask)
○整数型:RC          /* 返却値 */
  ▲ a
  ・RC ← 2          /* 返却値は2 */
  ▲ b
  ・RC ← 0          /* 返却値は0 */
  ・RC ← 1          /* 返却値は1 */
・return RC         /* RCを返却値として返す */
```

・空欄a：空欄aが真の場合は返却値に2が設定されているので，空欄aではData中の検査したビットが全て1であることを判断できる条件を考える。前述の説明にもあるように，検査するビットの値だけを取り出すためには，検査対象のデータ（Data）とマスクビット（Mask）との論理積を求める。その結果，検査したビットだけが1，それ以外のビットは全て0であったときに，検査したビットが全て1であったと判断できる。そして，それは結果がMaskと同じ値ということである。したがって，条件式は(Data & Mask) = Maskと表現でき，（ウ）が正解である。
・空欄b：空欄bが真の場合は返却値に0（検査した全てのビットが0），偽の場合は返却値に1（検査したビット中に0と1が混在）を設定している。前述の説明にもあるように，Mask と Data の論理積を求めると，検査対象のビットはそのままの値，対象外のビットは全て0になるので，検査対象のビットが全て0の場合の演算結果は"00000000"Bに，1が含まれる場合には，それ以外の値になる。したがって，空欄bの条件式は(Data & Mask) ="00000000"Bと表現でき，（ア）が正解である。

［設問2］
プログラム1では，Mask中に1のビットが1個以上あることを前提としていたが，この前提を取り除いて，Mask中の1のビットが0個の場合は，返却値0を返すようにプログラムを変更する。この変更のための修正案は①～③の三つあり，適切なものを選択する。なお，修正案中の空欄a，bは，それぞれ設問1で解答した“(Data & Mask) = Mask”，“(Data & Mask) = "00000000"B”が入ることに注意する。
Mask中の1のビットが0個の場合にDataとMaskの論理積を求めると，演算結果のビット列は全て0になるため，その判定条件は(Data & Mask) = "00000000"Bと表現できる。また，Mask自体のビット列が"00000000"Bであるため，(Data & Mask) = Maskとも表現できる。このことを基に修正案を順番に検証していく。

修正案①（変更なし）
```
▲(Data & Mask) = Mask
・RC ← 2
  ▲(Data & Mask) = "00000000"B
  ・RC ← 0
  ・RC ← 1
・return RC
```

修正案①では，Maskのビット列が全て0の場合，空欄a（(Data & Mask) = Mask）が真となるので，本来は返却値として0を返すべきところを2を返すことになり，正しい結果とはならない。

修正案②
```
▲(Data & Mask) = "00000000"B
・RC ← 0
  ▲(Data & Mask) = Mask
  ・RC ← 2
  ・RC ← 1
・return RC
```

修正案②では，最初の判定条件が“(Data & Mask) = "00000000"B”なので，Maskのビット列が全て0の場合には返却値が0となる。また，検査した全てのビットが0の場合もこの条件が成り立つので，返却値0に関する判定は正しく行うことができる。そして，この条件が偽の場合には，続いて“(Data & Mask) = Mask”の判定を行うが，その結果は，検査したビットが全て1の場合はMaskと同じビット列に，0と1が混在する場合はMaskと異なるビット列になるため，真のときに2，偽のときに1と正しい返却値を返している。したがって，正しく動作する。

修正案③
```
・RC ← 1
▲(Data & Mask) = "00000000"B
・RC ← 0
▲(Data & Mask) = Mask
・RC ← 2
・return RC
```

修正案③では，最初の判定条件である“(Data & Mask) = "00000000"B”が真となった場合にも，続けて“(Data & Mask) = Mask”の判定を行うことになる。修正案①の説明にもあるように，Mask中の1のビットが0個の場合にはこの二つの判定条件はどちらも真となるため，結果として返却値が2となってしまい，正しい結果とはならない。
以上の内容から，正しく動作するのは修正案②だけであることが分かり，空欄cは（イ）が正解である。

［設問3］
プログラム2，3の処理効率について考える。

［プログラム2］
```
（処理量）
        ○整数型関数:BitCount (8ビット論理型:Data)
        ○8ビット論理型：Work
        ○整数型:Count, Loop
①  1    ・Work ← Data
②  1    ・Count ← 0
③  4    Loop: 0, Loop < 8, 1
④  3      ▲Workの最下位ビットが1
⑤  1      ・Count ← Count + 1
⑥
⑦  1    ・Workを右へ1ビット論理シフトする
⑧
⑨  2    ・return Count          /* Countを返却値として返す */
```

※○付き数字は行番号を表す。

・空欄d：整数型関数BitCountは，8ビットのデータ中にある1のビットの個数を返す。プログラム2は，最下位ビットが1かどうかを判断することで1のビットの個数をカウントするものであるが，ビットの判定の都度，ビット列を1ビットずつ右へシフトすることで，ビット番号2，ビット番号3，…，ビット番号7を順番に最下位ビットへ移動させ，8ビット全てについての判定を行う。そのため，行番号③の繰返しの判断は9回（Loopは0から始まりLoop=8で繰返し処理を抜ける），行番号④，⑦の処理は8回実行される。また，行番号⑤の処理はWorkの最下位ビットが1の場合だけ実行されるので，実行回数が最小になるのはWorkのビットが全て0の場合，逆に最大になるのはWorkのビットが全て1の場合であり，実行回数はそれぞれ0回，8回となる。これを整理すると，最小，最大の処理量は次のようになる。

行番号	処理量	最小の処理量	最大の処理量
①	1	1×1回=1	1×1回=1
②	1	1×1回=1	1×1回=1
③	4	4×9回=36	4×9回=36
④	3	3×8回=24	3×8回=24
⑤	1	1×0回=0	1×8回=8
⑦	1	1×8回=8	1×8回=8
⑨	2	2×1回=2	2×1回=2
		合計　72	合計　80