文部科学省後援

情報システム試験

令和５年度前期 情報検定

＜実施　令和5年9月10日（日）＞

基本スキル

（説明時間　13：00～13：10）

（試験時間　13：10～14：10）

・試験問題は試験開始の合図があるまで開かないでください。

・解答用紙（マークシート）への必要事項の記入は，試験開始の合図と同時に行いますので，それまで伏せておいてください。
・試験開始の合図の後，次のページを開いてください。**＜受験上の注意＞**が記載されています。必ず目を通してから解答を始めてください。
・試験問題は，すべてマークシート方式です。正解と思われるものを１つ選び，解答欄の()をＨＢの黒鉛筆でぬりつぶしてください。２つ以上ぬりつぶすと，不正解になります。
・辞書，参考書類の使用および筆記用具の貸し借りは一切禁止です。
・電卓の使用が認められます。ただし，下記の機種については使用が認められません。

＜使用を認めない電卓＞

１．電池式（太陽電池を含む）以外の電卓
２．文字表示領域が複数行ある電卓（計算状態表示の一行は含まない）
３．プログラムを組み込む機能がある電卓
４．電卓が主たる機能ではないもの
　＊パソコン（電子メール専用機等を含む），携帯電話，スマートフォン，タブレット，電子手帳，電子メモ，電子辞書，翻訳機能付き電卓，音声応答のある電卓，電卓付き腕時計，時計型ウェアラブル端末等
５．その他試験監督者が不適切と認めるもの

一般財団法人　職業教育・キャリア教育財団

＜受験上の注意＞

１．この試験問題は12ページあります。ページ数を確認してください。
乱丁等がある場合は，手をあげて試験監督者に合図してください。
※問題を読みやすくするために空白ページを設けている場合があります。

２．解答用紙（マークシート）に，受験者氏名・受験番号を記入し，受験番号下欄の数字をぬりつぶしてください。正しく記入されていない場合は，採点されませんので十分注意してください。

３．試験問題についての質問には，一切答えられません。自分で判断して解答してください。

４．試験中の筆記用具の貸し借りは一切禁止します。筆記用具が破損等により使用不能となった場合は，手をあげて試験監督者に合図してください。

５．試験を開始してから30分以内は途中退出できません。30分経過後退出する場合は，もう一度，受験番号・マーク・氏名が記載されているか確認して退出してください。なお，試験終了５分前の合図以降は退出できません。試験問題は各自お持ち帰りください。

６．試験後の合否結果（合否通知），および合格者への「合格証・認定証」はすべて，Ｗｅｂ認証で行います。
①情報検定（J検）Webサイト合否結果検索ページ及びモバイル合否検索サイト上で，ディジタル「合否通知」，ディジタル「合格証・認定証」が交付されます。
②団体宛には合否結果一覧ほか，試験結果資料一式を送付します。
③合否等の結果についての電話・手紙等でのお問い合わせには，一切応じられませんので，ご了承ください。

問題１　次のテスト技法に関する記述を読み，各設問に答えよ。

システム開発における単体テストでは，プログラムの不具合を効率的に発見しなければならない。テスト技法として代表的なものに，プログラムの内部構造に着目してテストデータを作成し,プログラムの論理が正しいかどうかを検証する方法と,プログラムの外部仕様に着目し，入力データと出力結果だけを見て，機能と性能が要求どおりに動作しているかどうかを検証する方法がある。

＜設問１＞　次のテストに関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

［　(1)　］におけるテストデータの設計方法として，同値分割や限界値分析がある。例えば，入力項目が整数値の 20 以上～40 以下であるとき，同値分割では図１に示すように正常に処理される範囲の有効同値クラスと，異常と見なされる範囲の無効同値クラスに分けて各クラスの範囲内から最低でも１個を選びテストデータとする。図１の場合のテストデータは［　(2)　］である。限界値分析では有効同値クラスの境界になるデータをテストデータとする。図１の場合，有効同値クラスの下限の境界から［　(3)　］と上限の境界から［　(4)　］をテストデータとする。

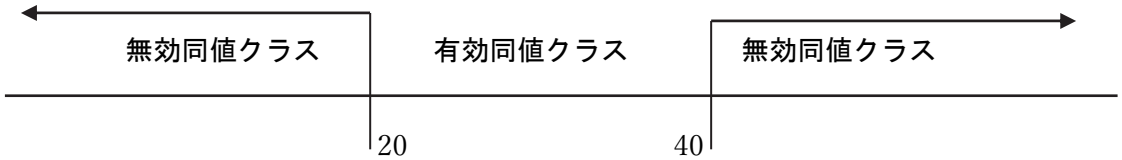


図１　テストデータ設計の例

(1) の解答群

ア．トップダウンテスト　　イ．ブラックボックステスト
ウ．ホワイトボックステスト　　エ．ボトムアップテスト

(2) の解答群

ア．10，19　　イ．10，20，40
ウ．10，30，50　　エ．41，50

(3)，(4) の解答群

ア．15，20　　イ．19，20
ウ．40，41　　エ．40，45

＜設問２＞　次のテストケースにおける網羅率に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

プログラムの内部仕様をもとにテストケースを設計する［　(5)　］では，テストケースとして表のような種類がある。テストを効率よく行うため，テストケース(又は経路など)のカバー率を表す網羅率を利用する。

表　テストケースの種類

種　類	説　明
命令網羅	すべての命令を少なくとも１回実行する
分岐網羅(判定条件網羅)	すべての分岐を少なくとも１回実行する
条件網羅	すべての条件の真と偽を少なくとも１回実行する
複数条件網羅	すべての条件で真と偽の組み合わせを実行する

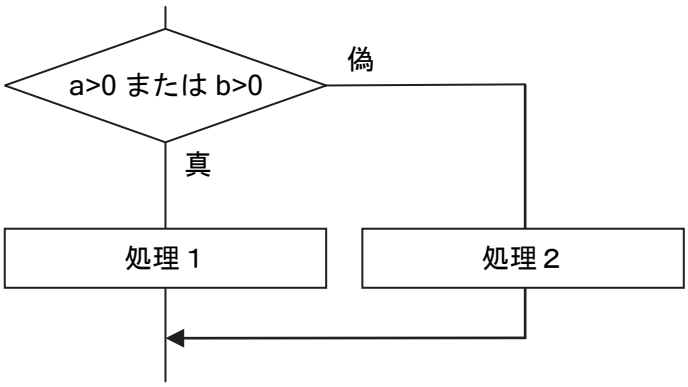


図２　分岐の流れ図の例

図２の流れ図において条件網羅で最低限のテストケースを設定すると，a についての判定および b についての判定がそれぞれ真と偽になる場合をテストすればよいので，(a=1，b=0)と(a=0，b=1)の２通りで十分である。しかし，これではすべての処理を実行しておらず，分岐網羅率は［　(6)　］%となる。

また，同様に最低限のテストケースを設定すると複数条件網羅のテストケースは［　(7)　］となり，命令網羅のテストケースは，［　(8)　］となる。

(5) の解答群

ア．システムテスト　　イ．ブラックボックステスト
ウ．ホワイトボックステスト　　エ．モジュール集積テスト

(6) の解答群

ア．0　　イ．25　　ウ．50　　エ．75

(7)，(8) の解答群

ア．(a=0，b=0)

イ．(a=0，b=0)と(a=1，b=1)

ウ．(a=0，b=0)と(a=0，b=1)と(a=1，b=0)と(a=1，b=1)

エ．(a=0，b=1)と(a=1，b=1)または(a=1，b=0)と(a=1，b=1)

問題２　次のシフト演算に関する記述を読み，各設問に答えよ。

シフト演算とは，ビット列を左または右へ移動する演算である。シフト演算には，論理シフト，算術シフト，循環シフトなどの種類がある。

論理シフトはビット列全体を移動するものである。移動によりあふれるビット列は無くなり，空いたビットには0が格納される。例えば，2進数0110を左へ２ビット論理シフトを行うと1000になる。

算術シフトは，符号ビットを除いたビット列を移動するものである。ここでは，符号ビットを最左端ビットとする。移動によりあふれたビット列は無くなるが，空いたビットには，左算術シフトの場合は0が，右算術シフトの場合は符号ビットと同じ値が格納される。例えば，2進数1100を左へ２ビット算術シフトを行うと1000になり，右へ２ビット算術シフトを行うと1111になる。

循環シフトは，ビット列全体を移動するものであるが，移動によりあふれたビット列は，空いたビットに順番に格納される。例えば，1001を右へ２ビット循環シフトを行うと0110になる。

＜設問１＞　次のシフト演算に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

８ビットの2進数11001100があり，この値に対して３ビットのシフト演算を行う。なお，各シフト演算はすべて単独で行うものとする。

右論理シフトを行った結果は［ (1) ］，左算術シフトを行った結果は［ (2) ］，左循環シフトを行った結果は［ (3) ］となる。

(1) ～ (3) の解答群

ア．00011001　　イ．01100000
ウ．01100110　　エ．10011001
オ．11100000　　カ．11111001

＜設問２＞　次のシフト演算によるビット列の取り出しに関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。なお，解答は重複して選んでよい。また，ここでのシフト演算は全て論理シフトとする。

左右の論理シフトを組み合わせることで任意のビット列を取り出すことができる。例えば，16進数ABCDの２桁目を取り出して0B00とすることを考える。まず16進数ABCDを左へ４ビットシフトを行ってBCD0とし，さらに右へ12ビットシフトを行えば000Bとなる。これを左へ８ビットシフトを行えば0B00になる。

同様に16進数ABCDの３桁目を取り出して00C0とする場合は，［ (4) ］ビットシフトして，次に［ (5) ］ビットシフトし，最後に左へ４ビットシフトする。

また，16進数ABCDの１桁目を取り出してA000とする場合は，［ (6) ］ビットシフトして，次に［ (7) ］ビットシフトすればよい。

(4) ～ (7) の解答群

ア．左へ４　　イ．左へ８　　ウ．左へ12
エ．右へ４　　オ．右へ８　　カ．右へ12

＜設問３＞　次のシフト演算を利用した乗算に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。なお，ここで行うシフト演算は全て算術シフトであり，桁あふれは発生しないものとする。

算術シフトを利用した乗算について考える。左へｎビットシフトを行うと元の値の2^n倍になるため，これを組み合わせて乗算ができる。組み合わせるシフト数は，乗数を2進数で表現したときに１が立っているビットの位置で決める。例えば，５倍するのであれば，５を2進数で表現すると101であるから，2^0と2^2の位に１が立っているので，シフトしていない元の値に元の値を左へ２ビットシフトした値を加えれば良い。

同様に，10倍するのであれば，元の値を左へ１ビットシフトした値に元の値を左へ３ビットシフトした値を加えれば良い。なお，シフトした結果をさらにシフトするようにすれば，元の値を左へ１ビットシフトした値に，さらにその結果を左へ２ビットシフトした値を加えることでも実現できる。ここで後者の考え方で，100倍するためには，元の値を左へ２ビットシフトした値に，さらにその結果を左へ［ (8) ］ビットシフトした値，さらにその結果を左へ［ (9) ］ビットシフトした３つの値を加える。

また，15倍するためには，元の値を［ (10) ］ことで計算できる。

(8)，(9) の解答群

ア．１　　イ．２　　ウ．３　　エ．４

(10) の解答群

ア．左へ１ビットシフトした値と，さらに左へ２ビットシフトした値を加える
イ．左へ２ビットシフトした値と，さらに左へ１ビットシフトした値を加える
ウ．左へ３ビットシフトした値と，さらに左へ１ビットシフトした値を加える
エ．左へ４ビットシフトした値から元の値を引く

問題３　次のデータ構造に関する記述を読み，各設問に答えよ。

データ構造とはデータがプログラムでどのように扱われるかを定義したものである。データ構造は，プログラムの処理効率に大きな影響を与えるため，様々な構造が考えられている。代表的なものにスタックとキューがある。

［スタックについて］

スタックとは後入れ先出し法(LIFO)によりデータを管理するメモリ領域のことである。プログラム中で副プログラムや関数を呼び出すときのアドレス情報などを一時的に蓄える場所として使われることがある。ここでは，スタックにデータを格納する場合は push，スタックからデータを取り出す場合には pop を使う。

・push　書式：push(データ)
　　例：　push(100)　　※スタックに 100 を格納する
・pop　書式：pop()
　　例：　x ← pop()　※スタックからデータを取り出して変数 x へ代入する

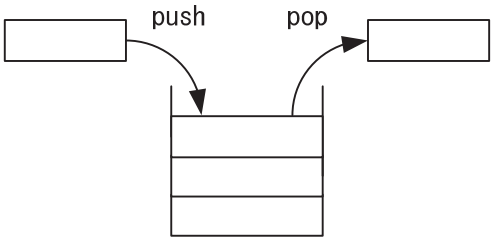


図１　スタックへのデータの出し入れ

［キューについて］

キューとは先入れ先出し法(FIFO)によりデータを管理するメモリ領域のことである。OS の待ち行列管理などに用いられる。ここでは，キューにデータを格納する場合は enq，キューからデータを取り出す場合は deq を使う。

・enq　書式：enq(データ)
　　例：　enq(100)　　※キューに 100 を格納する
・deq　書式：deq()
　　例：　x ← deq()　※キューからデータを取り出して変数 x へ代入する

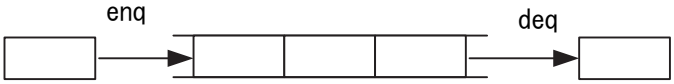


図２　キューへのデータの出し入れ

＜設問１＞　次のスタックとキューに関する記述を読み，記述中の［　　］に入れるべき適切な字句を解答群から選べ。

スタックの初期状態は図３である。なお，最後に格納されたデータは 50 である。

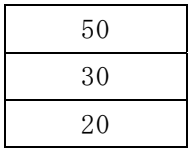


図３　初期状態のスタック

キューの初期状態は図４である。なお，最後に格納されたデータは 10 である。

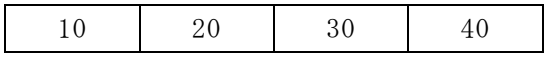


図４　初期状態のキュー

次の①～③の処理をそれぞれ初期状態のスタックとキューに対して実行した。

①
```
push(40)
x ← pop()
push(20)
x ← pop()
x ← pop()
```

②
```
x ← deq()
x ← deq()
enq(30)
enq(40)
x ← deq()
```

③
```
push(deq())
enq(pop())
x ← deq()
enq(pop())
x ← pop()
```

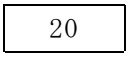
①の実行が終了したときの x の値は［(1)］，スタックの状態は［(2)］である。
②の実行が終了したときの x の値は［(3)］，キューの状態は［(4)］である。
③の実行が終了したときの x の値は［(5)］，スタックの状態は［(6)］，キューの状態は［(7)］である。

(1)，(3)，(5) の解答群

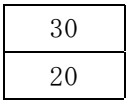
ア．10　　イ．20　　ウ．30　　エ．40　　オ．50

(2)，(6) の解答群

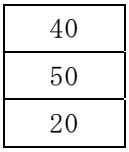
ア．



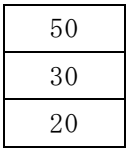
イ．



ウ．



エ．



(4)，(7) の解答群

ア.

30	40	10	50

イ.

40	30	10

ウ.

40	30	40

エ.

50	40	10	20

＜設問２＞　次の処理の説明を読み，記述中の［　　］に入れるべき適切な字句を解答群から選べ。

スタックおよびキューは設問１の初期状態から始める。また，処理中の「print」は，値を標準出力装置へ出力するための命令である。

次の処理を実行すると，［(8)］の順に出力される。

［処理］

```
push(20)
push(deq())
print deq()
enq(pop())
print pop()
enq(pop())
print deq()
print pop()
```

(8) の解答群

ア．10，20，20，50　　イ．20，20，30，30

ウ．30，20，20，30　　エ．30，20，30，50

問題４　次の CPU アーキテクチャに関する各設問に答えよ。

CPU を構成する制御装置内には様々なレジスタや回路がある。これらのレジスタや回路が連携して，メモリ上の命令を実行している。

＜設問１＞　次の命令実行手順に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

コンピュータで一つの演算をする場合の命令は，主に次の 6 つのステージ（過程）に基づいて実行される。

Ⅰ．これから実行する命令が主記憶装置のどこに記憶されているかを示すアドレスを記憶するためのレジスタである［(1)］（プログラムカウンタ）が示すアドレスを参照し，情報を取り出す。取り出した情報を制御装置内の［(2)］に格納する。

Ⅱ．［(2)］に格納された情報は命令部とアドレス部に分かれており，命令部の情報は［(3)］で解読され，使用すべき算術論理回路が演算装置に指示される。

Ⅲ．［(2)］のアドレス部の情報は［(4)］に送られ，操作する対象データが記憶されているアドレスが計算される。

Ⅳ．操作対象のデータが取り出され，演算装置に送られる。

Ⅴ．演算装置で命令が実行される。

Ⅵ．実行された演算結果が主記憶装置に記憶される。

このような流れで命令は実行され，まとめると［(5)］→命令実行→演算結果格納の順となる。

(1) ～ (4) の解答群

ア．アキュムレータ　　イ．アドレスレジスタ
ウ．ディスパッチャ　　エ．命令アドレスレジスタ
オ．命令デコーダ　　カ．命令レジスタ

(5) の解答群

ア．オペランドフェッチ→デコード→命令フェッチ→オペランドアドレス計算
イ．オペランドフェッチ→オペランドアドレス計算→デコード→命令フェッチ
ウ．命令フェッチ→デコード→オペランドアドレス計算→オペランドフェッチ
エ．命令フェッチ→オペランドアドレス計算→デコード→オペランドフェッチ

＜設問２＞　次のアドレス指定方式に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

CPU の命令が操作の対象とするデータのアドレスを実効アドレスといい，実効アドレスを指定する方式には様々なものがある。各レジスタ，主記憶装置の条件と内容が図のような時の処理を考える。

［条件］
命令レジスタのアドレス部：1000
指標レジスタ：100

番地	1000	1100	1200	1300	1400	1500	1600
データ	1200	1400	1100	1000	1600	1300	1500

図　主記憶装置

ここで，指標アドレス指定方式の場合，実効アドレスは［(6)］となる。さらに，その実効アドレスを使用し，直接アドレス指定方式で実行する場合，取り出されるデータは［(7)］となる。また，間接アドレス指定方式で実行する場合，取り出されるデータは［(8)］となる。

(6) ～ (8)の解答群

ア．1000　　イ．1100
ウ．1200　　エ．1300
オ．1400　　カ．1500
キ．1600　　ク．1700

問題５　次のタスク管理に関する記述を読み，各設問に答えよ。

オペレーティングシステム(OS)から見た仕事の単位をタスクという。OS は複数のタスクに対して順に CPU を割り当て，仕事を実行させる。

＜設問１＞　次のタスクの状態遷移に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

OS は，CPU を時分割に割り当てながら複数のタスクを同時並行的に実行させるマルチタスク機能を備えている。マルチタスク環境で複数のタスクの同時並行動作を実現するために，OS はタスクの生成から消滅までを，実行可能状態，実行状態，待ち状態の三つの状態で管理している。

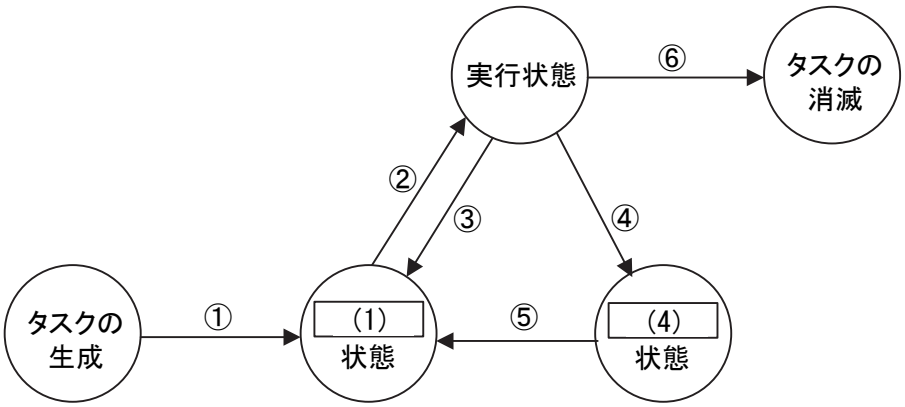


図　タスクの状態遷移

① 生成されたタスクは，使用する入出力装置が使用可能になると［　(1)　］状態になる。

② ［　(1)　］状態のタスクの中から実行するタスクを選択し，そのタスクに CPU の使用権が割り当てられ実行状態となる。この CPU の割り当てを［　(2)　］と呼ぶ。

③ 実行中のタスクは，タイマ割込みなどによって［　(1)　］状態へ遷移し，他のタスクが CPU を利用できるようになる。このように CPU の使用を一定時間とし，全てのタスクにできるだけ公平に CPU を割り当てるタスクスケジューリングを［　(3)　］方式と呼ぶ。

④ 実行状態中に，入出力要求が発生すると，タスクは［　(4)　］状態へ遷移する。このように入出力など OS の機能を利用するため，スーパバイザに依頼するのが SVC 割込みである。

⑤ タスクは入出力終了によって，待ち状態から［　(1)　］状態へ遷移する。このとき発生するのが入出力割込みである。

⑥ CPU の割り当てを繰り返し，処理を終了したタスクは消滅する。

(1) ～ (4) の解答群

ア．実行可能	イ．ディスパッチ	ウ．プリエンプション
エ．待ち	オ．優先度順	カ．ラウンドロビン

＜設問２＞　次のマルチプログラミングに関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

三つのタスクＡ，Ｂ，Ｃがあり，各タスクを単独で実行した場合の資源使用状況と使用時間を表に示す。CPU は 1 個であり，1 個の CPU は 1 コアで構成される。また I/O は競合せず，OS のオーバヘッドは考慮しないものとする。タスクスケジューリングは優先度順方式でタスクＡ，Ｂ，Ｃの順とするが，CPU 使用中に割り込むことはない。なお，三つのタスクは同時に投入されるものとする。

表　各タスク単独実行時の資源使用状況

タスクＡ	CPU(30)	I/O(40)	CPU(10)
タスクＢ	CPU(20)	I/O(40)	CPU(10)
タスクＣ	CPU(10)	I/O(40)	CPU(10)

()内は使用時間で単位はナノ秒

各タスクが到着してから終了するまでの時間（ターンアラウンドタイム）は，タスクＡが［(5)］ナノ秒，タスクＢが［(6)］ナノ秒，タスクＣが［(7)］ナノ秒である。タスクの到着からすべてのタスクの実行が終了するまでの，CPU の遊休時間は［(8)］ナノ秒である。

(5) ～ (8) の解答群

ア．20	イ．30	ウ．50	エ．60
オ．80	カ．100	キ．110	ク．120

＜メモ欄＞