〔C1 C2〕−2025− デジタル

専門（記述式）試験問題

注　意　事　項

1. 問題は **6 題**あります。このうち**任意の 2 題**を選んで解答してください。なお、この問題集の**裏表紙**に科目別構成の詳細が記載されていますので、解答開始前によく読んでおいてください。

2. 解答時間は **3 時間**です。

3. 答案用紙の記入について
   ㋐　答案は濃くはっきり書き、書き損じた場合は、解答の内容がはっきり分かるように訂正してください。
   ㋑　問題 **1 題に 1 枚**（両面）を使用してください。
   ㋒　表側の各欄にそれぞれ必要事項を記入してください。
   問題番号欄には、解答した問題の番号をそれぞれ記入してください。
   ㋓　試験の公正を害するおそれがありますので、答案用紙の氏名欄以外に氏名その他解答と関係のない事項を記載しないでください。

4. 下書き用紙はこの問題集の**中央部**にとじ込んであります。**試験官の指示**に従って、**試験開始後に**問題集から下書き用紙だけを慎重に**引きはがして**使用してください。なお、誤って問題集を破損しても、問題集の交換はできませんので注意してください。

5. この問題集で単位の明示されていない量については、全て国際単位系（SI）を用いることとします。

6. この問題集は、本試験種目終了後に持ち帰りができます。

7. 本試験種目の途中で退室する場合は、退室時の問題集及び下書き用紙の持ち帰りはできませんが、希望する方には後ほど渡します。別途試験官の指示に従ってください。なお、試験時間中に、この問題集から**下書き用紙以外**を切り取ったり、問題を転記したりしないでください。

8. 下欄に受験番号等を記入してください。

第 1 次試験地	試験の区分	受験番号	氏　　名
	デジタル		

指示があるまで中を開いてはいけません。

途中で退室する場合……本試験種目終了後の問題集の持ち帰りを　　**希望しない**

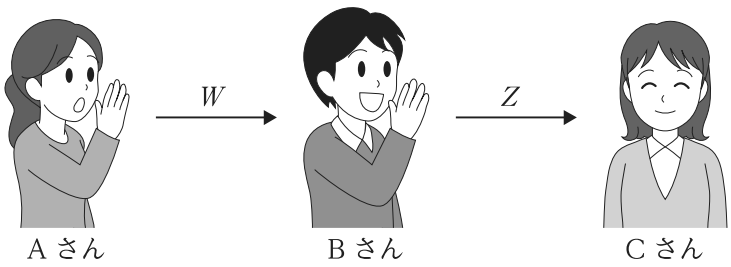
【No. **1**】 整数 a, $b(a \leqq b)$について、$[a, b]$ は a 以上 b 以下の整数の集合を表すものとする。N を 2 以上の整数として、A さん、B さん、C さんの 3 名で、以下のような手順(ア)、(イ)、(ウ)に従って伝言ゲームを行うとする。

(ア) A さんは、$[1, N]$ 中の整数を一つ、ある確率分布 D に従って選択し、それを B さんに伝える。

(イ) B さんは、A さんより伝えられた値を正整数 q で割った余りを C さんに伝える。

(ウ) C さんは、B さんより伝えられた値に基づき、A さんが選んだ値を推測する。

A さんが選択した値を確率変数 W、B さんが C さんへと伝えた値を確率変数 Z で表す。



X を値域が有限集合 $\mathcal{X}$ であるような離散確率変数とする。$a \in \mathcal{X}$ に対して、$X = a$ となる確率を $p_X(a)$で表す。確率変数 X のエントロピー $H(X)$は以下のように定義される。

$$H(X) = -\sum_{a \in \mathcal{X}} p_X(a) \log_2 p_X(a)$$

また、本問を通して、$0 \log_2 0 = 0$ と定義する。以下のⅠ、Ⅱの設問に答えよ。

Ⅰ．以下の問いに答えよ。

(1) $N = 10$, $q = 2$ として、D が一様分布であるとする。このときの $p_Z(0)$の値を求めよ。

(2) $N = 16$, $q = 3$ として、D が一様分布であるとする。このときの $H(W)$, $H(Z)$の値を求めよ。解答は対数値 $\log_2 3$、$\log_2 5$ を含む形で解答してよいものとする。

(3) $H(X)$が必ず非負の値となることを証明せよ。

(4) 次のような確率的試行により定まる $[1, N]$ 上の確率分布を考える。

試行はステップ 1 より開始される。i ステップ目$(1 \leqq i < N)$において、$[1, i+1]$ 中の値を一様分布に従って選ぶ。もし選んだ値が 1 であれば、値 i を選択して試行を終了し、そうでなければ $i+1$ ステップへと進む。N ステップ目に到達したときは値 N を選択する。

A さんがこの分布に従って W の値を決定するとしたとき、以下の問いに答えよ。

(a) $N > 3$ の場合における確率 $p_W(3)$ の値を求めよ。

(b) a, b を、$a < b < N$ を満たす正整数とする。$W \in [a, b]$ が成り立つ確率を a, b を用いて表せ。

Ⅱ. X, Y を、それぞれ値域が有限集合 $\mathcal{X}$, $\mathcal{Y}$ であるような離散確率変数とする。$a \in \mathcal{X}$, $b \in \mathcal{Y}$ に対して、$(X, Y) = (a, b)$ となる(結合)確率をそれぞれ $p_{XY}(a, b)$ で表す。また、$Y = b$ の下で $X = a$ となる条件付き確率(すなわち $p_{XY}(a, b)/p_Y(b)$ の値)を $p_{X|Y}(a|b)$ で表す。X の Y の下での条件付きエントロピー $H(X|Y)$、X と Y の相互情報量 $I(X, Y)$ をそれぞれ次のように定義する。

$$H(X|Y) = -\sum_{b \in \mathcal{Y}} p_Y(b) \sum_{a \in \mathcal{X}} p_{X|Y}(a|b) \log_2 p_{X|Y}(a|b)$$

$$I(X, Y) = \sum_{a \in \mathcal{X}, b \in \mathcal{Y}} p_{XY}(a, b) \log_2 \frac{p_{XY}(a, b)}{p_X(a) p_Y(b)}$$

以下の問いに答えよ。

(1) $N = 16$, $q = 2$ として、D が一様分布であるとする。このとき、$I(W, Z)$ を求めよ。

(2) $I(X, Y) = H(X) - H(X|Y)$ 及び $I(X, Y) = I(Y, X)$ が成立することを示せ。

(3) ある関数 $f: \mathcal{X} \to \mathcal{Y}$ により $Y = f(X)$ と表現できるとき、等式 $H(Y|X) = 0$ が成立することが知られている。条件付きエントロピーが表す情報量の直感的な意味を踏まえて、この式が成り立つ理由を説明せよ。

(4) D を任意の分布、q を任意の正整数とする。関数 $f: [0, q-1] \to \mathcal{X}$ を C さんの推定値を表す関数とする。このとき、以下の問いに答えよ。

ただし、Ⅰ(1)～Ⅰ(4)及びⅡ(1)、Ⅱ(2)、Ⅱ(3)において示している事実は証明なしに利用してよい。

(a) $H\big(f(Z)\big) \leqq H(Z)$ が成立することを示せ。

(b) $I\big(W, f(Z)\big) \leqq I(W, Z)$ が成立することを示せ。

【No. **2**】 クロック入力に伴って n 個の状態 $S_0, S_1, \ldots, S_{n-1}$ を循環的に遷移($S_0 \to S_1 \to \cdots \to S_{n-1} \to S_0 \to \cdots$)する順序機械を設計することを考える。以下のⅠ、Ⅱ、Ⅲの設問に答えよ。

ただし、次の条件に従うこととする。

・論理式に関しては、特に指示のない場合、積和形の最簡形で答えるものとする。
・D-FF 一つは 2 入力 NAND ゲート 6 個で構成されているものとする。
・D-FF d_i は、データ入力 D_i、クロック入力 C_i、データ出力 Q_i 及びデータ出力の否定出力 $\overline{Q_i}$ をもつとする。
・論理ゲートとして、NOT，AND，OR，NAND，NOR の 5 種類を使用可能とする。
・各論理ゲートの最大入力数は 3 とし、4 入力以上の多入力論理ゲートは使用できないものとする。
・各論理ゲートの構成に用いられている MOSFET の数は、以下の表に示すとおりであるとする。

NOT	2 入力 AND	3 入力 AND	2 入力 OR	3 入力 OR	2 入力 NAND	3 入力 NAND	2 入力 NOR	3 入力 NOR
2	6	8	6	8	4	6	4	6

Ⅰ．複数の D-FF から成る非飽和カウンタでこのような順序機械を実現する方法がある。$n = 6$ として六つの状態 $S_0, \ldots, S_5$ を循環的に遷移する順序機械を設計する場合、各状態は 3-bit の ID を割り当てることで区別できるため、三つの D-FF d_0，d_1，d_2 があればよい。以下の問いに答えよ。

⑴ 状態 $S_j (j = 0, 1, \ldots, 5)$ を、$Q_2Q_1Q_0$ が j の 2 進数 3 桁表現になるように割り当てるとする（例えば、$Q_2 = 0$，$Q_1 = 1$，$Q_0 = 1$ が状態 S_3 を表す。）。これは、すなわち、三つの D-FF で各 D-FF の初期値が 0 であるような 6 進(modulo-6)アップカウンタを実現することを意味する。このとき、各 D-FF に与える入力 D_i を生成する回路の論理式を、導出過程も含めてそれぞれ示せ。

⑵ 今度は、状態 S_j を、$Q_2Q_1Q_0$ が $(7 - j)$ の 2 進数 3 桁表現になるように割り当てるとする。これは、すなわち、三つの D-FF で各 D-FF の初期値が 1 であるような 6 進ダウンカウンタを実現することを意味する。このとき、各 D-FF に与える入力 D_i を生成する回路の論理式を、導出過程も含めてそれぞれ示せ。

⑶ ⑴で求めた論理式から、⑴のアップカウンタにおいて、D-FF 以外に必要となる論理ゲートを構成する MOSFET の総数を求めよ。また、⑵のダウンカウンタにおいても同様に求めよ。なお、各 D-FF は、否定出力 $\overline{Q_i}$ をもつことに注意すること。

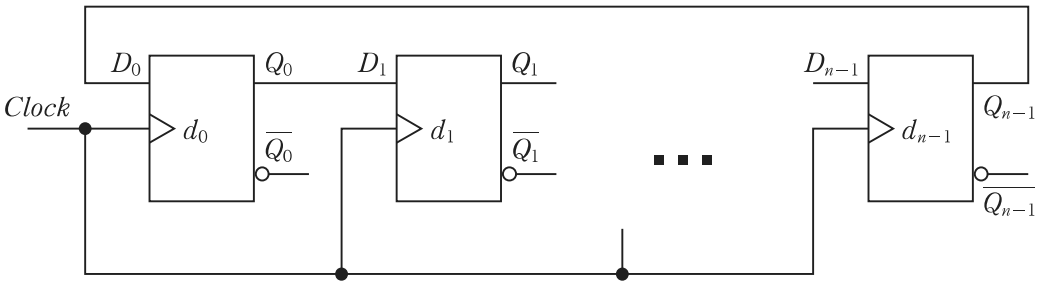
⑷　積和形で表現された論理式に基づいて実現した AND-OR 論理回路は、OR ゲートがもつ入力線それぞれに二重否定(直列接続された二つの NOT ゲート)を挿入した上で適切にゲートを置き換えることにより、NAND，NOR，NOT の３種のゲートのみ使用する回路に変換できる。⑵において求めた入力 D_1 を生成する回路の回路図及びそれを NAND，NOR，NOT の３種のみから成る回路に変換した後の回路図をそれぞれ示せ。

⑸　⑴、⑵で求めた論理式に基づく回路を全て、⑷で示した方法により NAND，NOR，NOT による回路に変換した場合において、⑶と同様に MOSFET の総数を、⑴のアップカウンタ及び⑵のダウンカウンタそれぞれについて求めよ。

ただし、変換前と変換後において、回路の入力として用いる D-FF の出力信号を変更してもよいものとする。

⑹　⑸の場合において、D-FF も含めた回路全体を構成する MOSFET の数を、⑴のアップカウンタ及び⑵のダウンカウンタそれぞれについて求めよ。

Ⅱ．n 状態を循環的に遷移する順序機械の他の実現方法として、図のような D-FF を単純にカスケード接続したリングカウンタがある。以下の問いに答えよ。



⑴　状態数 $n = 6$ としたとき、必要となる D-FF の数はいくらか。

⑵　⑴で求めた個数の D-FF により構成したリングカウンタが取り得る状態数は、各 D-FF の初期値によって異なる。取り得る状態数を全て挙げよ。

⑶　この方法で実現した、n 状態($n = 6$)を循環的に遷移する順序機械全体を構成する MOSFET の数を求めよ。

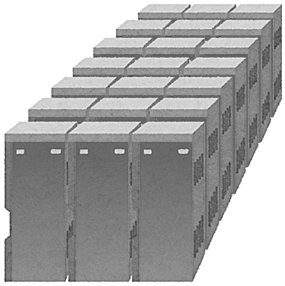
Ⅲ．リングカウンタにおいて、最上段の入力 D_0 に、最下段の出力 Q_{n-1} に代わり否定出力 $\overline{Q_{n-1}}$ を接続したものをジョンソンカウンタと呼ぶ。このジョンソンカウンタによっても、n 状態を循環的に遷移する順序機械を実現可能である。以下の問いに答えよ。

(1) 状態数 $n = 6$ としたとき、必要となる D-FF の数はいくらか。

(2) k 個の D-FF を使用したジョンソンカウンタで表現可能な状態数を k の式で示せ。

(3) この方法で実現した、n 状態($n = 6$)を循環的に遷移する順序機械全体を構成する MOSFET の数を求めよ。

【No.　3】 我が国の科学技術の発展を根底から支えてきたスーパーコンピュータについて、具体的な実例の概要を勉強しながら問題を解くことを考える。以下のⅠ～Ⅳの設問に答えよ。



Ⅰ．現代のスーパーコンピュータは、演算能力を高めるために、数万個以上の高性能 CPU がインターコネクトと呼ばれる高速ネットワークで相互接続された構造となっている。例えば、ある有名な国産スーパーコンピュータの諸元はおよそ以下のようなものである。

ただし、実物と比べて若干簡略化されている。

・CPU コアは 2 GHz のクロックで動作
・CPU コアは 1 クロックサイクル当たり最大八つの浮動小数点演算を実行可能
・CPU チップは 8 コア構造であり、CPU チップごとに十分なサイズのメインメモリが接続されている。これにさらにインターコネクトの機能を実現するためのインターコネクトコントローラ(ICC)チップも追加された「計算ノード」が計算を実行する基本単位となる。
・システムボードに四つの計算ノードを搭載
・各筐体(きょうたい)に 24 枚のシステムボードを実装
・幅 50 m、奥行 60 m の計算機室に、筐体を横方向に 24 個、奥行方向に 36 個配置
・計算ノードどうしは、6 次元メッシュ／トーラスと呼ばれる結合トポロジーのインターコネクトで結合され、計算ノードを経由してバケツリレー式でデータパケットをやり取りしながら、並列計算が行われる。

以下の問いに答えよ。

⑴　スーパーコンピュータが 1 秒間に実行することのできる浮動小数点演算の最大回数を「理論ピーク性能」と呼び、Flops(Floating point operations per second)という単位で表現する。上記のスーパーコンピュータの理論ピーク性能はいくらか。メガ、ギガなどの適切な接頭語を使って、有効数字 3 桁で求めよ。

⑵　プログラムを逐次実行した場合の全実行ステップ数のうち、並列処理によって実行を加速することが可能な部分の割合を「並列化率」と呼ぶ。上記のスーパーコンピュータで、並列化率が

99.99 % となるプログラムを実行させたとすると、プログラムの実行時間は最大でどのくらいの割合まで短縮されるか。また、このときの実効性能は、理論ピーク性能と比べてどのくらいの割合か。

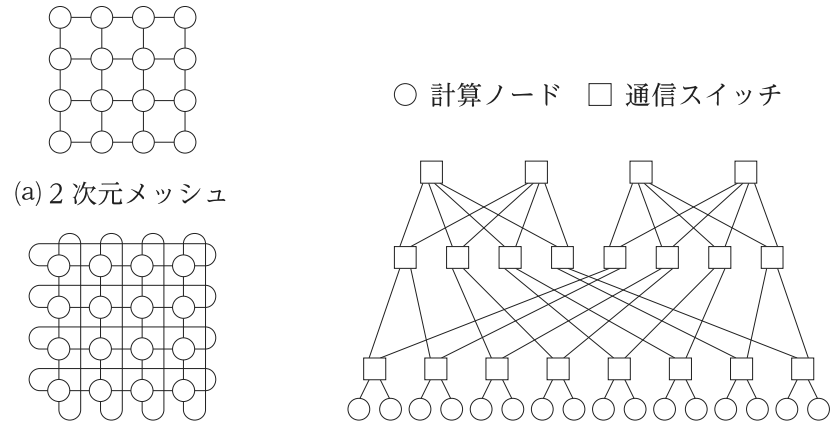
ただし、ここでは並列処理に伴う様々なオーバヘッドは無視できるものとする。

⑶ ⑵でも見たように、スーパーコンピュータで科学技術計算プログラムを実行させる場合、必ずしも常に理論ピーク性能を発揮しながら計算が行えるとは限らない。上記のようなスーパーコンピュータにおいて、実効性能を理論ピーク性能から乖離させてしまうと考えられる原因を三つ挙げ、どのように理論ピーク性能の実現が阻害されるかを、それぞれ 100 字以内で説明せよ。

⑷ 計算ノード当たりの消費電力を 145 W、一般家庭 1 世帯の平均消費電力を 400 W とすると、このスーパーコンピュータをフル稼働させたときの消費電力は何世帯分の電力に相当するか。

⑸ 上記のスーパーコンピュータは既に 1 世代前のものである。現代の最新のスーパーコンピュータの場合、主に近年出現した新たな応用への対応から、コストパフォーマンスや電力効率を高めつつ計算性能を補強するために、上記に加えて新たにもう一つ別の汎用部品を計算ノードに追加する場合も多くなっている。その汎用部品の名称と、それが主にどのような応用に対応するためのものか説明せよ。

Ⅱ．多数の計算ノードの間を結合して高速なデータのやり取りを可能とし、計算ノードどうしが協力し合って効率的に並列計算を行えるようにするために、これまで、様々な結合トポロジーのインターコネクトがスーパーコンピュータに採用されてきた。次の図Ⅰは、代表的な結合トポロジーの例である。



⒜ 2 次元メッシュ　⒝ 2 次元トーラス　⒞ファットツリー（4 ポートスイッチを使用した例）

図Ⅰ．代表的な結合トポロジーの例

任意のポート間での通信が可能な「通信スイッチ」をツリー状に接続すると、複数のスイッチを経由することで、任意のノード間で通信させることが可能となる。しかし、単純なツリー構造のままであると、上層になるほど通信容量(ケーブル本数)が減り、そこが通信のボトルネックとなってしまう。そこで、ファットツリー(図Ⅰの(c))では、スイッチごとにポートの半分を使って下層からのケーブルを受け取りつつ、残りのポート数の分だけ上層のツリーノードを重複して用意し、残りのポートからのケーブルをこれらのツリーノードにそれぞれ接続する。こうして、上層に行くほど多重化された(ファットな)ツリーを構成することにより、ボトルネックが生じないような工夫がなされている。

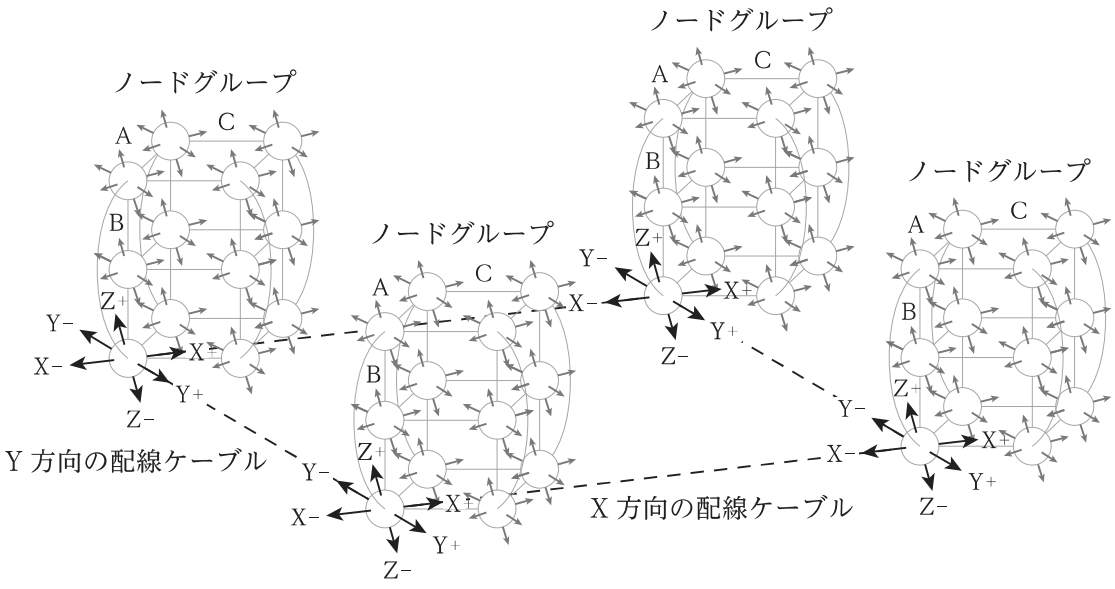
Ⅰで示したスーパーコンピュータよりも前の世代の、計算ノード数が数百～数千の規模のスーパーコンピュータでは、インターコネクトのトポロジーとしてファットツリーが用いられる場合が多かった。一方、計算ノード数が1万～10万超のスーパーコンピュータの世代になると、3次元メッシュ若しくは3次元トーラスなどの3次元接続を用いることが普通となってきている。以下の問いに答えよ。

(1) 計算ノード 10×10 台を1m間隔で2次元に配置し、なるべくケーブル長が均一となるようにしつつ2次元トーラスで接続した場合、最も長いケーブルの長さはいくらか。

(2) 8192台の計算ノードを32ポートのスイッチを使ってファットツリートポロジーで接続するために必要なスイッチの台数とケーブルの本数はそれぞれいくらか。

(3) インターコネクトのトポロジーの主流がファットツリーから3次元メッシュ若しくは3次元トーラスなどの3次元接続へと変わったのはどのような理由によるものと考えられるか。必要となるハードウェア量の観点から300字程度で説明せよ。

Ⅲ. Ⅰで示したスーパーコンピュータでは、インターコネクトとして、次元の大きさが6で、それぞれの次元ごとにメッシュあるいはトーラスのどちらかの構造をもつ「6次元メッシュ／トーラス構造」が採用されている。図Ⅱに示すように、計算ノード間の接続は、X，Y，Z，A，B，Cの六つの次元をもち、「ノードグループ」と呼ばれる大きさ $2 \times 3 \times 2$ のABC3次元メッシュ／トーラスネットワークが構成単位となり、これに含まれる各ノードどうしが、それぞれXYZ3次元メッシュ／トーラスで結合された構造をもつ。各計算ノードには、A，B，C軸接続用の4本の通信ポートに加えてX，Y，Z軸の +/− 方向の6本の通信ポートがあり、これらの6本の通信ポートを通じてX，Y，Z方向で隣接するノードグループ内の対応する位置の計算ノードと接続されている。



図Ⅱ．6 次元メッシュ／トーラスの構成要素となるノードグループ

A 軸と C 軸のメッシュ接続は、四つの計算ノードが載ったシステムボード内で結線される。さらに、3 枚のシステムボード上の計算ノード間を、筐体内でボードどうしを接続するバックプレーンで結線された B 軸でトーラス接続することで、12 ノードから成るノードグループが作られる。それぞれ 24 枚のシステムボードが実装された筐体二つ(Y 方向に背中合わせ)に収められた合計 16 のノードグループ間をリング接続する座標が Z 軸である。最後に、X 方向にトーラス接続、Y 方向にメッシュ接続され、これがシステム規模に対応する。以下の問いに答えよ。

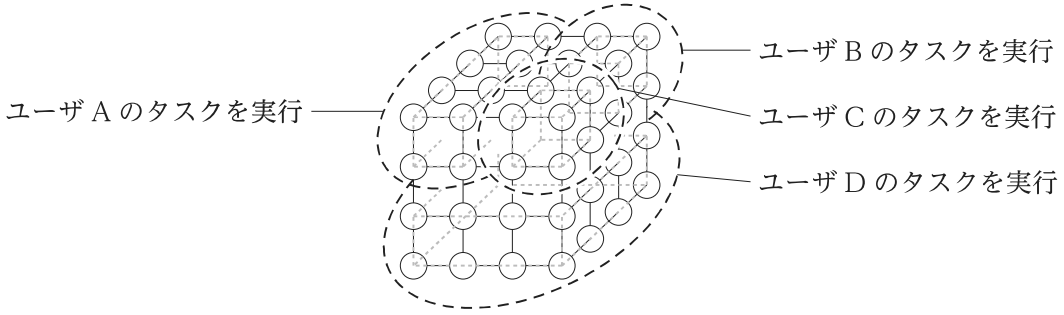
⑴ インターコネクトの XY 方向に接続するために、一つの筐体から出されているケーブルの本数はいくらか。

⑵ Ⅰで示したスーパーコンピュータは、各筐体に 24 枚のシステムボードが実装され、筐体が X 方向に 24 個、Y 方向に 36 個配置されている。X 方向の配線に用いられるケーブルの本数は全部でいくらか。

Ⅳ．Ⅰで示したスーパーコンピュータともなると、常に、最大構成システム全体を一人のユーザが独占して使用するとは限らない。実行されるプログラムの規模にあわせて全体システムを分割し、複数のユーザが同時に使用できるようになっていれば、スーパーコンピュータを最大限無駄なく利用することができる。

Ⅰで示したスーパーコンピュータには、インターコネクト全体を複数のより小さな 3 次元メッシュへと分割して独立に動作させる機構が用意されている。モデルとした実際のスーパーコンピュータではメッシュでなく仮想的なトーラスへ分割されるが、本問では簡単のためメッシュ

とした。図Ⅲのように、各タスクは、それが実行される３次元メッシュのサイズがユーザにより指定され、指定されたサイズに分割された個別の３次元メッシュで実行される。



図Ⅲ．４×４×４のメッシュを分割して複数のユーザタスクを同時実行する例

Ⅰで示したスーパーコンピュータのインターコネクトは、図Ⅱのような６次元メッシュ／トーラス構造となっているため、ユーザタスクを実行する３次元メッシュは、同じ接続関係を保った、より高次元の構造へと折りたたまれた上で、６次元メッシュ／トーラス構造の一部分へと埋め込まれる。高次元になるほど折りたたみ方に様々な可能性が生まれ、埋め込みの柔軟性が高まり、より多くのユーザの要求を同時に満たせる余地が増える。こうして、Ⅰで示したスーパーコンピュータの同時利用の効率を向上させている。以下の問いに答えよ。

⑴　まず、２次元メッシュと３次元メッシュでの埋め込みの柔軟性を比較することを考える。サイズ４×９の２次元メッシュで結合された計算ノード群があり、部分的に長方形を切り出してタスクを割り当てて実行することができるものとする。このような２次元メッシュでサイズ６×３のユーザタスクを実行させる場合、同時に実行することのできる最大タスク数はいくらか。一方、同じ数の計算ノードが、サイズ４×３×３の３次元メッシュで結合されており、そこから部分的に直方体を切り出してタスクを割り当てて実行できるものとすると、先と同じサイズ６×３のユーザタスクを、同じ接続関係を保った直方体へと折りたたんで３次元メッシュに埋め込んだ場合に、同時に実行することのできる最大タスク数はいくらか。

⑵　サイズ 12 × 12 × 6 の３次元メッシュのユーザタスクを、６次元の直方体へと折りたたんで、図Ⅱの６次元メッシュ／トーラス構造上で実行することを考える。折りたたんで作られる６次元直方体のサイズとしてあり得るものを全て挙げよ。

ただし、向きの異なるものは別のものとみなす。

【No. 4】 以下の設問に答えよ。

単一の変数では表現できない、桁数の多い数を表現したり計算したりすることがある。このとき、一つの数を配列で表現する。本問では、桁数の多い負でない整数(多倍長整数)の計算、特に乗算を考える。実装においてはC言語(ANSI C、C89)を用いるものとし、表現においては配列の一つの要素で10進法での1桁を表すものとする。要素番号kの要素に10^kの桁の数字を記憶する。例えば、大きさ6のintの配列a[]で「1732」を表すときは、a[0]は2、a[1]は3、a[2]は7、a[3]は1、a[4]は0、a[5]は0、となる。

ただし、本問においては、計算の途中において配列の各要素がオーバフローを起こすことは考えなくてよく、計算の開始前と終了後に前述の表現になっていればよいものとする。

また、プログラムにおいては、<stdlib.h> が読み込まれているものとし、malloc関数は失敗しないものとする。

⑴ 多倍長整数の乗算は、最も単純な考え方では、筆算と同じ要領で行える。これについて以下の問いに答えよ。

⑷ n桁の多倍長整数どうしの積を求める関数biMul1，biArrを次のように作成した。biArr関数は、多倍長整数を前述の所定の形式に整える関数である。㋐～㋓に適切な式を入れよ。

ただし、a[]、b[]、及びc[]の記憶領域の重複は考えなくてよい。また、ANSI Cでは負の整数の除算は処理系依存であることに注意すること。

```
void biMul1(int c[], const int a[], const int b[], int n) {
  int i, j;

  for (i = 0; i < n + n; i++) c[i] = 0;
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) c[ ㋐ ] += a[i] * b[j];
  }
  return;
}

void biArr(int a[], int n) {
  int cry = 0, i;

  for (i = 0; i < n; i++) {
    cry = (a[i] += cry) >= 0 ? ㋑ : ㋒ ;
```

```
    a[i] = [ ㊦ ] ;
  }
  return;
}
```

⑹ この方法で N 桁どうしの乗算を行うときの最悪時間計算量のオーダとして最も妥当なのを次の①～⑤から選べ。

ただし、当てはまるものが複数あるときは、最も小さいオーダを示すこと。

① $O(N\sqrt{N})$　② $O(N^2)$　③ $O(N^2 \log N)$　④ $O((N \log N)^2)$　⑤ $O(N^2\sqrt{N})$

⑵ ⑴の多倍長乗算に対し、計算量を抑えられるカラツバ法が知られている。

p 進法で 2 桁以内の負でない整数 a、b を、$a = a_1p + a_0$、$b = b_1p + b_0$（a_0、a_1、b_0、及び b_1 は 0 から $p - 1$ の整数）と表す（これは、n 桁の多倍長整数を、上位 n/2 桁分と下位 n/2 桁分に分割することに対応する。）。すると、$ab = (a_1p + a_0)(b_1p + b_0) = a_1b_1p^2 + (a_1b_0 + a_0b_1)p + a_0b_0 = a_1b_1p^2 + \{(a_1 + a_0)(b_1 + b_0) - a_1b_1 - a_0b_0\}p + a_0b_0$ と表せる。ここで a_0b_0、a_1b_1 は 1 度計算すればよいから、乗算回数は、a_1b_1、a_1b_0、a_0b_1、及び a_0b_0 を計算する 4 回から、a_1b_1、$(a_1 + a_0)(b_1 + b_0)$、及び a_0b_0 を計算する 3 回に減っている。これを再帰的に実行すれば乗算回数が抑えられる。これについて以下の問いに答えよ。

⑹ カラツバ法で、n 桁（ただし、n は 2 のべき乗；2, 4, 8, 16, ……）の多倍長整数どうしの積を求めるための関数 biMul2 を次のように作成した（また、⑴の biArr 関数も使用する。）。㋐～㋒に適切な式を入れよ。

ただし、⑴⒜と同様に、a[]、b[]、及び c[]の記憶領域の重複は考えなくてよい。

```
void biMul2(int c[], const int a[], const int b[], int n) {
  int m = n / 2, i;
  const int *a0 = a, *a1 = a + m, *b0 = b, *b1 = b + m;
  int *c0 = c, *c1 = c + n, *c2, *s1, *s2;

  if (n == 1) {
    c[0] = [ ㋐ ] ; c[1] = 0;
    return;
  }
  c2 = malloc(sizeof(int) * n);
  s1 = malloc(sizeof(int) * m); s2 = malloc(sizeof(int) * m);
  for (i = 0; i < [ ㋑ ] ; i++) {
```

```
    s1[i] = a1[i] + a0[i];
    s2[i] = b1[i] + b0[i];
  }
  biMul2(c0, a0, b0, m);
  biMul2(c1, a1, b1, m);
  biMul2(c2, [キ] , [ク] , m);
  for (i = 0; i < n; i++) c2[i] -= [ケ] ;
  for (i = 0; i < n; i++) c[ [コ] ] += c2[i];
  free(s1); free(s2);
  free(c2);
  return;
}
```

(b)　カラツバ法で N 桁どうしの乗算を行う時間計算量を $T(N)$ とする。$T(N)$ は乗算演算子による乗算回数 $T_{\mathrm{M}}(N)$ と加減算演算子による加減算回数 $T_{\mathrm{A}}(N)$ を用いて $T(N) = T_{\mathrm{M}}(N) + T_{\mathrm{A}}(N)$ と表せる。明らかに、$T_{\mathrm{M}}(1) = 1$、$T_{\mathrm{A}}(1) = 0$ である。

(i)　$T_{\mathrm{M}}(2N)$ を、$T_{\mathrm{M}}(N)$ を用いた式で表せ。また、$T_{\mathrm{A}}(2N)$ を、$T_{\mathrm{A}}(N)$ を用いた式で表せ。

(ii)　$T(N)$ の最悪のオーダ(カラツバ法の最悪時間計算量のオーダ)として最も妥当なのを次の①～⑩から選べ。

ただし、当てはまるものが複数あるときは、最も小さいオーダを示すこと。

① $O(N^{1.1})$　② $O(N^{1.2})$　③ $O(N^{1.3})$　④ $O(N^{1.4})$　⑤ $O(N^{1.5})$

⑥ $O(N^{1.6})$　⑦ $O(N^{1.7})$　⑧ $O(N^{1.8})$　⑨ $O(N^{1.9})$　⑩ $O(N^{2.0})$

(3)　カラツバ法をさらに進めた多倍長乗算アルゴリズムに Toom-Cook 法がある。カラツバ法では多倍長整数を二つに分割していたが、ここでは三つに分割する Toom-Cook-3 法を考える。

p 進法で 3 桁以内の負でない整数 a、b を、$a = a_2p^2 + a_1p + a_0$、$b = b_2p^2 + b_1p + b_0$ (a_0、a_1、a_2、b_0、b_1、及び b_2 は 0 から $p - 1$ の整数)と表す(これは、n 桁の多倍長整数を、上位 n/3 桁分、中位 n/3 桁分、及び下位 n/3 桁分に分割することに対応する。)。ここで、これらの a, b の表現を p の多項式とみれば、$a = A(p)$, $b = B(p)$ とおける。積 ab は $A(p)B(p)$ であり、これも p の多項式である。これを $C(p)$ とおく。ここで、Toom-Cook-3 法では、$C(p) = c_4p^4 + c_3p^3 + c_2p^2 + c_1p + c_0$ とも表せるから、p として異なる五つの値を代入して得られる式と式の値とで連立方程式を作り、係数 c_0、c_1、c_2、c_3、及び c_4 を定められる。

係数 c_0、c_1、c_2、c_3、及び c_4 を定めるに当たり、簡単のため、p として、式の値が求め

やすい -2、-1、0、1、及び2を代入すると、ある行列 X を用いて $\begin{pmatrix} C(-2) \\ C(-1) \\ C(0) \\ C(1) \\ C(2) \end{pmatrix} = X \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}$ が得られる。いま求めたいのは c_0、c_1、c_2、c_3、及び c_4 なので、必要なのは X の逆行列 $X^{-1} = \frac{1}{24}\begin{pmatrix} 0 & 0 & 24 & 0 & 0 \\ 2 & -16 & 0 & 16 & -2 \\ -1 & 16 & -30 & 16 & -1 \\ -2 & 4 & 0 & -4 & 2 \\ 1 & -4 & 6 & -4 & 1 \end{pmatrix}$ である。この X^{-1} は c_0、c_1、c_2、c_3、及び c_4 の具体的な値によらない(すなわち、a にも b にもよらない。)、Toom-Cook-3法を使うときに共通に使用できる(プログラム中に組み込む)行列である。

さらに、$C(p) = A(p)B(p)$ であることを用いる。先ほどと同様に、p として -2、-1、0、1、及び2を代入すれば、$A(-2) = 4a_2 - 2a_1 + a_0$、$A(-1) = a_2 - a_1 + a_0$、$A(0) = a_0$、$A(1) = a_2 + a_1 + a_0$、及び $A(2) = 4a_2 + 2a_1 + a_0$ と表せる。$B(p)$ についても同様の表現が得られる。また、$C(p) = A(p)B(p)$ であるから、$C(-2) = A(-2)B(-2)$、$C(-1) = A(-1)B(-1)$、$C(0) = A(0)B(0)$、$C(1) = A(1)B(1)$、及び $C(2) = A(2)B(2)$ である。よって、$C(-2)$、$C(-1)$、$C(0)$、$C(1)$、及び $C(2)$ をそれぞれ $A(-2)B(-2)$、$A(-1)B(-1)$、$A(0)B(0)$、$A(1)B(1)$、及び $A(2)B(2)$ によって求め、定数倍して加えれば、c_0、c_1、c_2、c_3、及び c_4 が求められ、これが p 進法の各桁の値であることに立ち返れば積 ab を構成できる。

これらを再帰的に実行し、カラツバ法よりさらに乗算回数を抑えた多倍長乗算が実現できる。これについて以下の問いに答えよ。

(a) 問題文中の行列 X を成分表示で示せ。

(b) Toom-Cook-3法で、n桁(ただし、nは3のべき乗；3, 9, 27, 81, ……)の多倍長整数どうしの積を求めるための関数 biMul3 を次のように作成した(また、(1)の biArr 関数も使用する。)。ここで、［オ］には(2)(a)と同一の式が入る。㋐～㋖に適切な式を入れよ。ここで、㋐～㋒には順に $A(-2)$、$B(-2)$、$A(-1)$、$B(-1)$、$A(0)$、$B(0)$、$A(1)$、$B(1)$、$A(2)$、及び $B(2)$ を求める式が入る。また、㋓～㋗には順に c_0、c_1、c_2、c_3、及び c_4 を求める式が入る。

ただし、乗除算の回数を減らしたいため、㋐～㋗においては乗除算を使用してはならない。また、(1)(a)、(2)(a)と同様に、a[]、b[]、及びc[]の記憶領域の重複は考えなくてよい。

```
void biMul3(int c[], const int a[], const int b[], int n) {
  int m = n / 3, i;
  const int *a0 = a, *a1 = a + m, *a2 = a + (m + m);
```

```
const int *b0 = b, *b1 = b + m, *b2 = b + (m + m);
int *c0 = c, *c2 = c + (m + m), *c4 = c + (m + m + m + m);
int *c1, *c3;
int *an2, *an1, *a00, *ap1, *ap2, *bn2, *bn1, *b00, *bp1, *bp2;
int *cn2, *cn1, *c00, *cp1, *cp2;

if (n == 1) {
   c[0] = [ ㋕ ] ; c[1] = 0;
   return;
}
c1 = (int *)malloc(sizeof(int) * (m + m));
c3 = (int *)malloc(sizeof(int) * (m + m));
an2 = (int *)malloc(sizeof(int) * m);
an1 = (int *)malloc(sizeof(int) * m);
a00 = (int *)malloc(sizeof(int) * m);
ap1 = (int *)malloc(sizeof(int) * m);
ap2 = (int *)malloc(sizeof(int) * m);
bn2 = (int *)malloc(sizeof(int) * m);
bn1 = (int *)malloc(sizeof(int) * m);
b00 = (int *)malloc(sizeof(int) * m);
bp1 = (int *)malloc(sizeof(int) * m);
bp2 = (int *)malloc(sizeof(int) * m);
cn2 = (int *)malloc(sizeof(int) * (m + m));
cn1 = (int *)malloc(sizeof(int) * (m + m));
c00 = (int *)malloc(sizeof(int) * (m + m));
cp1 = (int *)malloc(sizeof(int) * (m + m));
cp2 = (int *)malloc(sizeof(int) * (m + m));
for (i = 0; i < m; i++) {
  an2[i] = [ ㋖ ] ; bn2[i] = [ ㋗ ] ;
  an1[i] = [ ㋘ ] ; bn1[i] = [ ㋙ ] ;
  a00[i] = [ ㋚ ] ; b00[i] = [ ㋛ ] ;
  ap1[i] = [ ㋜ ] ; bp1[i] = [ ㋝ ] ;
  ap2[i] = [ ㋞ ] ; bp2[i] = [ ㋟ ] ;
}
biMul3(cn2, an2, bn2, m);
```

（下書き用紙）

（下書き用紙）

（下書き用紙）

（下書き用紙）

```
    biMul3(cn1, an1, bn1, m);
    biMul3(c00, a00, b00, m);
    biMul3(cp1, ap1, bp1, m);
    biMul3(cp2, ap2, bp2, m);
    for (i = 0; i < m + m; i++) {
      c0[i] = [ ㋐ ] ;
      c1[i] = ( [ ㋑ ] ) / 12;
      c2[i] = ( [ ㋒ ] ) / 24;
      c3[i] = ( [ ㋓ ] ) / 12;
      c4[i] = ( [ ㋔ ] ) / 24;
    }
    for (i = 0; i < m + m; i++) {
      c[ [ ㋕ ] ] += c1[i];
      c[ [ ㋖ ] ] += c3[i];
    }
    free(cn2); free(cn1); free(c00); free(cp1); free(cp2);
    free(an2); free(an1); free(a00); free(ap1); free(ap2);
    free(bn2); free(bn1); free(b00); free(bp1); free(bp2);
    free(c1); free(c3);
    return;
  }
```

(c) (2)(b)と同様に考えて、Toom-Cook-3 法で N 桁どうしの乗算を行う最悪時間計算量のオーダとして最も妥当なのを次の①～⑩から選べ。

ただし、当てはまるものが複数あるときは、最も小さいオーダを示すこと。

① $O(N^{1.1})$　② $O(N^{1.2})$　③ $O(N^{1.3})$　④ $O(N^{1.4})$　⑤ $O(N^{1.5})$

⑥ $O(N^{1.6})$　⑦ $O(N^{1.7})$　⑧ $O(N^{1.8})$　⑨ $O(N^{1.9})$　⑩ $O(N^{2.0})$

(d) 一般に、Toom-Cook-k 法では、多倍長整数を k 分割して乗算回数を $2k-1$ 回まで減らす。すると、k が大きくなると時間計算量は減少し、桁数 N について $O(N)$ に近づいていくはずである。しかし、k が大きい Toom-Cook 法は速度面で実用にならないことが知られている。その理由を 2 行以内で説明せよ。

【No. 5】 多数のデータから成る集合があるときに、その集合に属さないある一つのデータが与えられたとき、それに最も近いデータを集合の中から探す最近傍探索は、多くの空間情報処理・パターン認識・ロボティクスのタスクで必要となる処理である。以下のⅠ、Ⅱ、Ⅲの設問に答えよ。

Ⅰ．最近傍探索は、データの集合を D 次元ユークリッド空間 $\mathbb{R}^D$ の点の集合 $P = \{\boldsymbol{p}_i, i = 1 \ldots N\}$ として表し、データ $\boldsymbol{x}$, $\boldsymbol{y}$ 間に距離 $d(\boldsymbol{x}, \boldsymbol{y})$ が定義されているとき、データの集合 P の中からクエリ $\boldsymbol{q}$ に対して最も近接するデータ $\mathrm{NN}(\boldsymbol{q}, P)$ を探してくる問題であり、$\mathrm{NN}(\boldsymbol{q}, P) \equiv \underset{\boldsymbol{p}_i \in P}{\operatorname{argmin}}\, d(\boldsymbol{q}, \boldsymbol{p}_i)$ と定式化できる。

(1) データを D 次元空間の点としてその座標を要素数 D のリストで表し、データの集合を要素数 N のそのリストで表す。ユークリッド距離 $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$ を用いる場合、最も単純な Python コードは次のようになる。以下の問いに答えよ。

ただし、Python のバージョンは 3.9、**はべき乗、float('inf')は無限大を意味し、各行の先頭の数字は行番号とする。

```
01  def dis(x, y):
02      return sum([(xi - yi)**2 for xi, yi in zip(x, y)])** 0.5
03
04  def brute_force_nn(p, q):
05      min_d, min_p = float('inf'), None
06      for pi in p:
07          d = dis(pi, q)
08          if d < min_d:
09              min_d, min_p = d, pi
10      return min_d, min_p
```

(a) p=[[50,20], [90,30], [30,80], [70,90], [10,50], [20,90], [80,60]]、q=[30,70] のとき、関数 brute_force_nn(p,q) の返り値を示せ。

(b) このアルゴリズムの計算量を、距離を計算する関数 dis の呼び出し回数で評価するときの最悪の計算量のオーダとして最も妥当なのを次の①～⑤から選べ。

ただし、当てはまるものが複数あるときは、最も小さいオーダを示すこと。

① $O(\sqrt{N})$　② $O(N)$　③ $O(N^2)$　④ $O(\log N)$　⑤ $O(N \log N)$

⑵　この探索を高速化するために木構造を用いることを考え、次のようなコードを作った。以下の問いに答えよ。

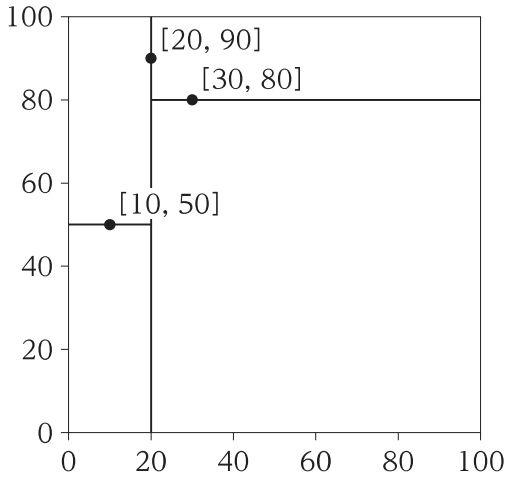
```
01  class Node:
02    def __init__(self, split, left, right):
03        self.split = split
04        self.left = left
05        self.right = right
06
07  def build_tree(p, depth=0):
08    if not p:
09      return
10
11    k = len(p[0])
12    axis = depth % k
13
14    p.sort(key=lambda x:x[axis])
15    med = len(p) // 2
16    print(axis, med, p[med])
17
18    node = Node(
19      p[med],
20      build_tree(p[0:med], depth+1),
21      build_tree(p[med+1:], depth+1)
22    )
23    return node
```

⑷　⑴⑷のpを用いbuild_tree(p)としたとき、表示される文字列を示せ。

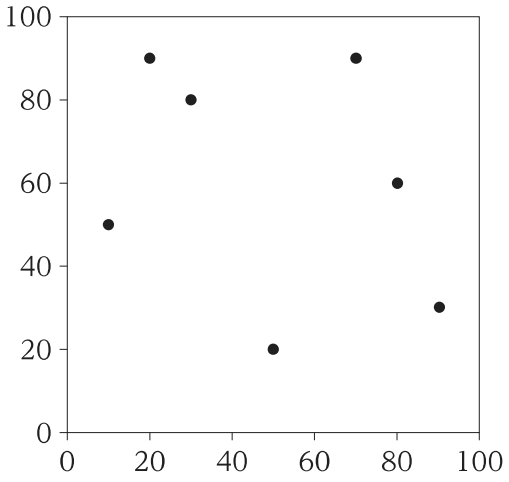
⑹　14行目の処理の内容を1行程度で説明せよ。

⑹　この木構造は、データの各点をノードとして空間を分割して探索を高速化している。空間は、点群pから生成した木構造によって分割される。例えば、⑴⑷のpの点のうち[20,90], [30,80], [10,50]だけ使用して生成した木構造を図Ⅰのように示すこととする。pは図Ⅱのように分布している。このとき、⑴⑷のpの全ての点を使用して生成した木構造を描け。

ただし、答案用紙に図Ⅱを描き写し、そこに各点の座標を示すこと。



図Ⅰ



図Ⅱ

(3) この木構造を用いて最近傍点を探索する Python コードを次のように作成した。以下の問いに答えよ。

```
00  def query_tree(tree, q, depth=0):
01      global min_d, min_p, bounds, done
02
03      k = len(tree.split)
04      axis = depth % k
05
06      if depth == 0:
07          min_d, min_p = float('inf'), None
08          bounds = [ [float('-inf'), float('inf')] for _ in tree.split ]
09
10      d = dis(q, tree.split)
11      if d < min_d:
12          min_d, min_p = d, tree.split
13
14      if tree.left is not None:
15          t = bounds[axis][1]
16          bounds[axis][1] = tree.split[axis]
17          query_tree(tree.left, q, depth+1)
18          bounds[axis][1] = t
```

```
19
20    if tree.right is not None:
21        t = bounds[axis][0]
22        bounds[axis][0] = tree.split[axis]
23        query_tree(tree.right, q, depth+1)
24        bounds[axis][0] = t
```

この関数を次のように呼び出す。ここで、(2)の関数 build_tree も使用する。

```
p=[[50,20], [90,30], [30,80], [70,90], [10,50], [20,90], [80,60]]
kdt = build_tree(p)
q=[30,70]
query_tree(kdt, q)
```

(a) どの順番でデータを探索し、最終的にどのような min_d, min_p の値が得られるか説明せよ。

(b) 木構造の探索方法として深さ優先探索と幅優先探索があるが、これはどちらの探索方法か。

(c) 次のような関数を定義する。

```
def bounds_intersect_region(q, bounds, r):
    return sum([
        (max([b[0] - xi, 0]) + max([xi - b[1], 0]))** 2
        for xi, b in zip(q, bounds)
    ]) <= r * r
```

さらに、コードの行番号 17 と 23 をそれぞれ次の行番号 17a, 17b と 23a, 23b のように書き換える。行番号 17a と 23a のインデントは、それぞれ行番号 17 と 23 のインデントと同じとする。

```
17a       if bounds_intersect_region(q, bounds, min_d):
17b           query_tree(tree.left, q, depth+1)

23a       if bounds_intersect_region(q, bounds, min_d):
23b           query_tree(tree.right, q, depth+1)
```

(i) (1)(a)の p を用いた場合、距離を計算する関数 dis の呼び出し回数は何回か。

(ii) 関数 bounds_intersect_region は何を判定しているか説明せよ。

(iii) このコードの変更はどのような効果をもたらすか 3 行程度で説明せよ。

(d) 次のような関数 in_region を定義する。なお、Python において要素を論理値とするリストに関数 min を適用した結果は要素全部の論理積になる。

```
def in_region(q, bounds, r):
    return min([
        b[0] + r <= xi <= b[1] - r
        for xi, b in zip(q, bounds)
    ])
```

さらに、コードの行番号 09, 13, 14, 20 をそれぞれ次のコード 09a, 13a, 13b, 14a, 20a に書き換える。書き換え部分の最初の行のインデントは元の行のインデントと同じとする。変数 done は False で初期化されているとする。

```
09a   if done: return

13a   if tree.left is None and tree.right is None:
13b         done = in_region(q, bounds, min_d)

14a   if not done and tree.left is not None:

20a   if not done and tree.right is not None:
```

(i) (1)(a)の p を用いた場合、距離を計算する関数 dis の呼び出し回数はいくらか。

(ii) 関数 in_region は何を判定しているか説明せよ。

(iii) このコードの変更はどのような効果をもたらすか 3 行程度で説明せよ。

(4) 以下の問いに答えよ。

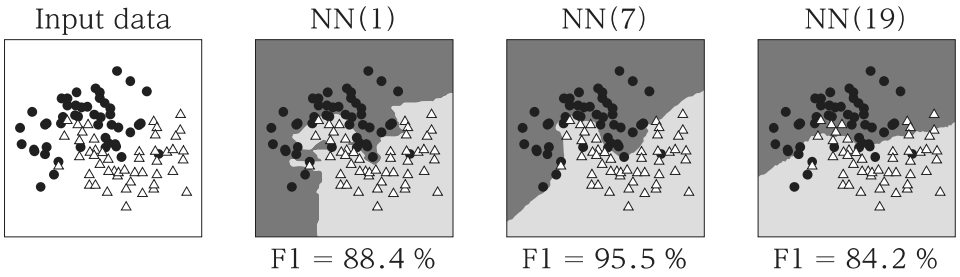
(a) 次の文章の㋐、㋑に当てはまるものを(1)(b)の選択肢から選べ。

ただし、当てはまるものが複数あるときは、最も小さいオーダを示すこと。

「関数 dis の呼び出し回数を N とするとき、この探索方法は二分木なので、平均的な時間計算量は [㋐] であるが、最悪の時間計算量は [㋑] になる。」

(b) 最悪な場合の例として、二次元の場合、データが円周上に分布していて、クエリがその円の中心付近にある場合がある。その場合に探索の効率が悪くなる理由を 3 行程度で説明せよ。

Ⅱ．最近傍探索を用いたデータの識別問題を考える。k-近傍法(k-nearest neighbor 法)は、ラベルが付けられた既知のデータの集合があるときに、クエリから k 番目に近いデータまでの距離を半径としたクエリを中心とする超球内にあるデータに付けられたラベルの多数決によって、そのクエリのラベルを決定する識別方法である。次の図Ⅲは二つのクラス(●と△で表示されているそれぞれ 50 点)について、そのうち 60 % を学習に使用し、k の値を 1，7，19 と変えて k-近傍法を適用し(NN(1)，NN(7)，NN(19))、その識別領域を図示したものである。残りの 40 % をテストデータとして識別した結果を真値と比較した F1 スコアをそれぞれの識別結果の下に示している。この図から、k の値の変化に伴って k-近傍法の識別能力に起こる変化とその理由について 3 行程度で述べよ。



図Ⅲ

Ⅲ．以下の問いに答えよ。

(1) パターン認識では、一般に高い次元のデータを扱うことが多い。データの次元と最近傍探索の効率との関連を考える。データが存在する D 次元ユークリッド空間 $\mathbb{R}^D$ での半径 r の超球を考える。超球の体積が $V_D(r) = A_D r^D$ (A_D は D に依存する定数であり、例えば、$D = 3$ のとき、$A_D = \dfrac{4}{3}\pi$ である。)として記述できるとする。以下の問いに答えよ。

(a) 半径 1 の超球と半径 $1 - \epsilon (0 \leqq \epsilon \leqq 1)$ の超球に挟まれた空間の体積が、半径 1 の超球の体積に占める割合はどのような式で記述できるか示せ。

(b) 横軸に ϵ、縦軸に(a)の割合をとったとき、どのようなグラフになるか、$D = 1, 2, 5, 10$ の場合について概形を重ねて描け。

(c) 次の文章は最近傍探索の効率が次元 D によってどのような影響を受けるか説明している。㋒、㋓、㋔に適切な値又は語句を入れよ。

「D が大きくなると、(a)で求めた関数はとても小さな ϵ に対しても急速に［ ㋒ ］に近づくようになる。つまり高次元空間では、超球内のほとんどの体積は超球表面に［ ㋓ ］部分で占められ、クエリからデータへの距離は全てほとんど［ ㋔ ］値になってしまう。」

(d) (c)の結果、データの次元 D が大きい場合に最近傍探索アルゴリズムはどのような困難に直面することになるか、3行以内で説明せよ。

【No. **6**】 ソフトウェアのテストに関する以下の設問に答えよ。

(1) 次の文章を読み、以下の問いに答えよ。

「ソフトウェア開発では様々な成果物を作るが、それらを作ったら終わりではなく、作られた成果物を確認する作業も重要である。この確認には、検証と妥当性確認という２種類が必要とされる。検証は開発のあるフェーズで作られた成果物が、フェーズの開始時に課せられた条件や制約を満たしているかを評価するものである。一方、妥当性確認は［ ① ］。

このような成果物の２種類の確認を行うためには様々な方法が用いられるが、その一つがテストであり、プログラムを実際に動かして確認するものである。実行により、期待通りの機能を果たすか、実行時の品質が期待どおりであるか、といったことを確認するのである。なお、実行せずに確認するための各種の技術についても静的テストと呼ばれることがあるが、本問ではテストとは実際に動かして確認するもののみを指すこととする。

テストはいくつかの観点で分類することができるが、その一つがテストレベルである。特定のプログラム、クラス、コンポーネントなどを他の部分と切り離して実行しその部分の欠陥を発見するための［ ㋐ ］、コンポーネント等の部分を統合してそれらの間のインタフェースと相互処理に関する欠陥を発見するための［ ㋑ ］、システム全体が指定された要件を満たすか確認する［ ㋒ ］、そのシステムが稼働でき目的を達成できるものであるか、典型的には顧客が判断を行う［ ㋓ ］がある。

別の観点としてはテストタイプがあり、機能が仕様どおり実装されているかを確認する機能テスト、機能以外の非機能的な側面を確認する非機能テスト、ソフトウェアが変更された際に行われる再テストや回帰テストがある。

また、テストによる確認をシステムの内部構造に基づいて行うものを［ ㋔ ］と呼び、内部構造に基づかずに仕様ベースで行うものを［ ㋕ ］と呼ぶ。

テストを行う際には、入力値、実行条件、［ ㋖ ］の集合である［ ㋗ ］を複数用意する。テスト対象のシステムやコンポーネントのために［ ㋗ ］をまとめたものを［ ㋘ ］と呼ぶ。」

(a) ㋐～㋘に当てはまる最も妥当なものを、それぞれ次の語句の中から選び出して示せ。

［語句：受入れテスト、期待される出力値、検証テスト、最終テスト、システムテスト、実行条件、第１テスト、第２テスト、第３テスト、妥当性確認テスト、単純テスト、単体テスト、テストケース、テストスイート、テストデータ、統合テスト、ブラックボックステスト、分岐テスト、ホワイトボックステスト、命令］

(b) ①には、妥当性確認とは何かを説明する文章が入る。１行程度で説明せよ。

(c) テストレベルとテストタイプは、どのような観点によるテストの分類であるか。それぞれ１～２行程度で説明せよ。

(d)　回帰テストとは何か、２行程度で説明せよ。

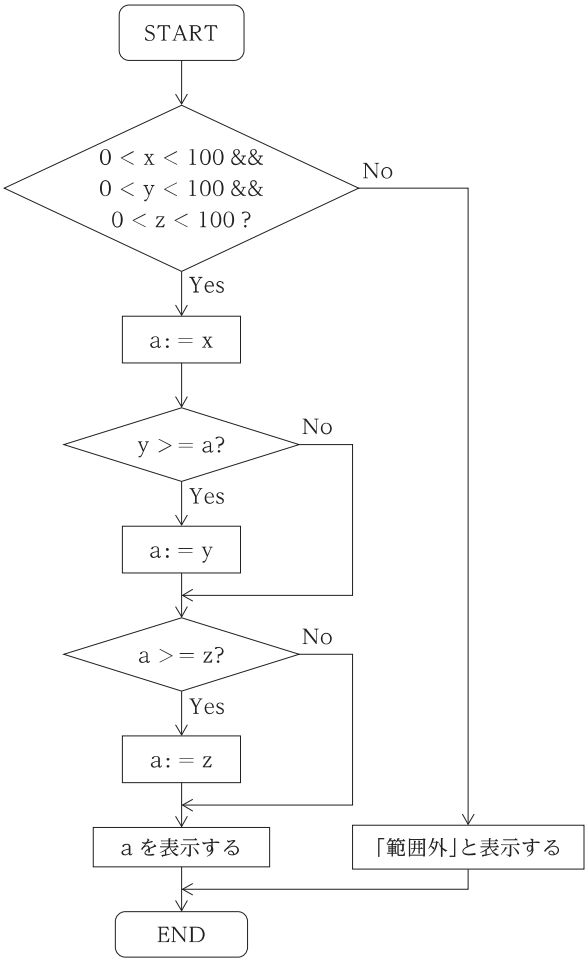
(e)　ソフトウェアの非機能的な側面には、実際に動かして確認するテストでは確認できないものがある。そのうちの一つを答えよ。また、テストでは確認できない理由と、それを確認するためのテスト以外の方法について、合わせて４行程度で説明せよ。

(2)　システムの内部構造に基づいて行うテスト((1)の㋕)について、テストケースの網羅率を考える。図は、ある人が作成した関数func(x, y, z)の制御構造をプログラムどおりに示したものである。この関数は、以下の仕様を満たすように作成された。

・０以上100以下である三つの数が入力されるとその最大値を表示

・入力範囲外の数値が入力されれば「範囲外」と表示

ただし、この仕様を満たすように正しく実装されているかどうかは分からないものとする。作成したこの関数のテストについて考える。以下の問いに答えよ。



(a)　この関数のテストケース①として入力値(x, y, z) = (10, 20, 20) で１回テストしたとき、カバレッジ基準を命令としたときの網羅率(命令網羅率)と、カバレッジ基準を分岐としたときの網羅率(分岐網羅率)はそれぞれ何％になるかを、その根拠とともにそれぞれ４行程度で説明せよ。

(b)　この関数についてさらに別の入力値のテストケース②を用意してもう１回テストすると、①、②を使ったテストにおける(a)のそれぞれの網羅率は、(a)の一つのテストケースによるテストのときと比較してどうなるか。次のA、B、Cから選択して答えよ。また、そうなる理由をテストケース②の入力値の例を挙げながら６行以内で説明せよ。

A　必ず上がる。

B　必ず上がるとは言えないが、下がることはない。

C　下がる場合もある。

(c)　それぞれのカバレッジ基準において網羅率が 100 % となるようなテストケース群のうち、テストケース数が最小となる例を一つずつ示せ。なお、各テストケースは、その入力値のみを示せばよい。

(d)　実は作成したこの関数には間違いが含まれている。以下の問いに答えよ。

(i)　その間違いは何かを指摘せよ。なお、間違いは複数含まれている可能性がある。

(ii)　さらに、その間違いは、命令網羅率、分岐網羅率が共に 100 % となるようなテストケース群であればどのテストケース群を用いても、必ず発見することができるかどうかを、その理由とともに４～５行程度で説明せよ。

ただし、間違いが複数ある場合には、それぞれについて４～５行程度で説明すること。

(3)　システムの内部構造に基づかないテスト((1)のㇵ)について、テストケース群をどのように作成すればよいかを考える。

次に示す仕様のシステムのテストについて、以下の問いに答えよ。

・会員が専用サイトで購入を行うと、１回の購入ごとに、購入金額、会員種別(ノーマル会員、ゴールド会員、プラチナ会員のいずれか)に応じて、以下のルールでポイントを付与
・購入金額が 10,000 円以上の場合は、500 ポイント
・プラチナ会員には、購入金額にかかわらず 200 ポイントを追加
・プラチナ会員、ゴールド会員には、購入金額が 10,000 円以上の場合に上記のルールで付与されたポイントに加えて 100 ポイントを追加

(a)　このシステムを仕様に基づいてテストすることを考える。仕様通りに作成されていることを確認するために効果的なテストケース群をどのように設計すればよいか、設計方法を 10 行程度で説明せよ。必要に応じて図や表を用いて説明を補足してもよい(図表は説明の行数には入れない。)。

ただし、入力値は、１以上の整数値で表現された購入金額、購入者の会員種別を示す文字(ノーマル会員はＮ、ゴールド会員はＧ、プラチナ会員はＰ)、出力は付与ポイント数とする。また、購入金額として１以上の整数値以外は入力できず、購入者の会員種別についてＮ、Ｇ、Ｐ以外は入力できないことは保証されているとしてよい。

(b)　(a)の設計方法に基づいて作成したテストケース群を示せ。

(4)　Ａさん、Ｂさん、Ｃさんは、あるソフトウェアのテストに関わっている。以下は、この３人がテストに際してどのような決定をしたかを記述したものである。この３人の決定のうち、**妥当でない**と思われるものを、その理由とともに答えよ。妥当でないと思われるものが複数あれば、それぞれについて答えよ。

・Ａさんは、モジュールＰのテストを担当することになった。モジュールＰの機能Ｆはモジュールｌの呼び出しを必要とする。よって機能Ｆに関するテストは統合テストに持ち越す

こととし、モジュールＰの単体テストからは除外した。

・Ｂさんは、サブシステムＳのテストを担当することになった。Ｓは重要な機能を含むサブシステムであり、入力値の範囲も広く様々な条件判断を含むものである。網羅的なテストが必要だと考え、全数テストをすることにした。

・Ｃさんは、モジュールＭのテストを担当することになった。予定していたテスト項目を30％消化した時点でプログラムの重大なバグが見つかったため、その修正を実施した。修正完了後、直ちに未消化のテスト項目を実施することにした。

科目別構成の詳細

科　　目	出題数	問題番号	ページ
計算機科学	1題	No. **1**	1～2
情報工学(ハードウェア)	2題	No. **2**, **3**	3～10
情報工学(ソフトウェア)	2題	No. **4**, **5**	11～23
情報技術	1題	No. **6**	24～27

○　**6題**のうちから**任意の2題**を選んで解答してください。