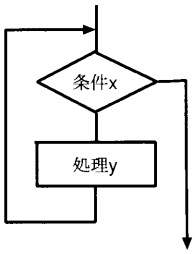
長崎大学大学院生産科学研究科 (博士前期課程) **受験番号 _____**

電気情報工学専攻 (情報システム工学系)

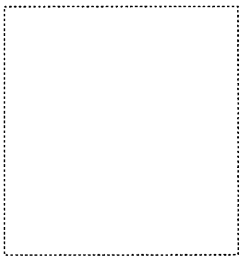
平成 20 年度 入学試験問題 ソフトウェア

1. 次の文章について，以下の問いに答えよ．

手続き型のプログラミング言語においては，プログラムに記述された順に逐次的に処理が実行される．このとき，処理の流れを制御するしくみとして重要となるのが (ア) と (イ) である．(ア) は，条件が成り立つか否かを判定して処理の内容を切り替えるしくみであり，C 言語では (ウ) や (エ) を用いて記述される．(イ) は，条件が成り立つ限り処理を何度も実行するしくみであり，以下の 2 通りのフローチャートが考えられる．すなわち，処理 y を実行する前に条件 x を判定する手順 (フローチャート A) と，処理 y を実行した後に条件 x を判定する手順 (フローチャート B) である．C 言語で フローチャート A の構文に用いられるのは (オ)，(カ) であり，フローチャート B の構文に用いられるのは (キ) である．



(フローチャート A)



(フローチャート B)

一方，関数の内部で自分自身を再び呼び出すことによってある処理を何度も実行することができる．これを (ク) 処理という．たとえば次のプログラムは，実数 b の n 乗を計算するプログラムであるが，関数 power の内部から自分自身を呼び出すことによって，実数 b を連続して n 回掛け合わせる処理が実行される．

```
#include <stdio.h>

double power(double b, int n);

int main(void)
{
    double b=5.0, val;
    int n=3;

    val = power(b, n);

    printf("%lf \n", val);

    return 0;
}

double power( double b, int n)
{
    [          ]

}
```

(1) 上の文章の空欄 (ア)〜(ク) に最も適した語句を以下の選択肢から選んで解答欄に記入せよ．ただし，同じ記号の空欄には同じ選択肢が入る．

選択肢：

代入，再帰，分岐，脱出，終了，呼び出し，繰り返し，関数，define 文，continue 文，break 文，switch 文，do 文，if 文，goto 文，for 文，return 文，while 文，do-while 文，exit 文，call 文

(2) フローチャート B を解答欄に記入せよ．

(3) 上記の関数 power の空欄を埋めてプログラムを完成させよ．

解答欄：

(1)

ア		イ		ウ		エ	
オ		カ		キ		ク	

(2)

(3)

```
double power(double b, int n)
{
    [                                    ]
}
```

2. 基本ソフトウェアに関する以下の文章中の空欄 (1〜14) を埋めよ．なお [n.] で表した空欄に関してはもっとも適切な語句を示し (語句の一部が既に記入されているものもある)，{ n. a:…, b:…, c:… } で表した空欄に関してはその中の選択肢 (a,b, c など) から適切なものを選び，その記号で解答すること．

小規模なビルディングブロックからのプログラムの構成を実現するため，関数やモジュールといった処理・制御の抽象化・定義能力の提供は近代的な高水準言語における必須の機能である．関数の呼び出しは，

i. 呼び出し側での引数の組み立て

ii. サブルーチン呼び出しに対応する機械語命令の実行

iii. 局所変数を使った関数本体の命令の実行

iv. return 文に対応する機械語命令の実行

という一連の流れによって実行される．

まず，関数が引数を取る場合には，i. で呼び出し側で与えられる式 (これを **{ 1. a: 実引数，b:仮引数 }** と言う) を実行 (評価) し，結果を **{ 2. a:スタック，b: ヒープ }** またはレジスタなどに積む．厳密な仕様を持つ手続型言語の多くでは，この式の評価は **{ 3. a:計算結果の非決定性を避けるために左から右へと，b: CPU を効率よく利用するため並列に }** 行なう．全ての式の評価が行なわれたら，ii. によりジャンプを行なう．ジャンプ先は **{ 4. a:仕様設計時，b:コンパイル時，c:実行時 }** にその関数に割り当てられたアドレスである．PC(これは [**5.**] の頭文字を取ったものである) を書きかえる無条件ジャンプ命令，条件ジャンプ命令とは違い，ここで使う機械語は関数本体から復帰できるように **{ 6. a:ジャンプ先の番地の次のアドレス，b: ジャンプ元の番地の次のアドレス，c:プロセススケジューラのアドレス }** を記録するような処理も実行する命令となっている．

続いて iii. すなわち関数本体の実行が始まる．まず，引数以外の **{ 7. a: ポインタ型の変数，b: 大域変数，c: 局所変数 }** のための領域を **{ 8. a:コード領域 ，b: データ領域，c: ヒープ領域，d: スタック領域 }** 上に確保する．また変数のスコープをネストできる言語においては親の環境へのポインタを更新する．これにより全ての変数のアドレスが確定するので機械語命令列を正しく実行することができるようになる．計算の制御流が最後まで到達した場合には iv. を行ない，局所的に割り当てたメモリ領域を解放し，呼び出し側に戻ってくる．

このアドレスに基づく呼び出し元→呼び出し先モデルで説明できないものとしては，システムコール，RPC，メソッド呼び出しなどがある．システムコールでは呼び出し元と呼び出し先とが同じ [**9.**] にないなどの理由により ii. においてジャンプ先のアドレスを直接指定することが不可能である．そこで [**10. 命令**] によって OS 内の関数を間接的に呼び出している．RPC では呼び出し元と呼び出し先とが別のコンピュータ上にあるためアドレス指定だけでは呼び出すことができない．現在のインターネットで接続されたコンピュータ間であれば，少なくともコンピュータ自身の指定のために，4Byte(または新しい規格では 8Byte) の [**11. アドレス**] の指定が必要である．メソッド呼び出しは [**12. 言語**] と呼ばれる言語属が持つ機能である．同じ関数名，同じ型の引数の並び，同じ返値の型の「関数」が複数存在する可能性があり (これをメソッドの [**13. オーバー**] と言う)，しかもそのうちのどれを呼び出せばよいかがコンパイル時には決らない (これは変数に対して [**14. 束縛**] が可能であるからである) ため，ii. の直前に呼び出し先の決定のための計算が必要となる．

長崎大学大学院生産科学研究科 (博士前期課程)　　受験番号 _____
電気情報工学専攻 (情報システム工学系)
平成 20 年度 入学試験問題 ソフトウェア

解答欄

1.	2.	3.
4.	5.	6.
7.	8.	9.
10. 命令	11. アドレス	12. 言語
13. オーバー	14. 束縛	

3. (1) 以下の Java プログラムを UML クラス図として表現せよ.

```
interface Countable {
    public int numMatch (int x);
}

abstract class IntStorage {
    protected int value;
}

class BiTree extends IntStorage implements Countable {
    private Countable left;
    private Countable right;
    public BiTree(int v, Countable l, Countable r) {
        value = v;
        left = l;
        right = r;
    }
    public int numMatch (int x) {

       [1]

    }
}

class TriTree implements Countable {
    private Countable left;
    private Countable center;
    private Countable right;
    public TriTree(Countable l, Countable c, Countable r) {
        left = l;
        center = c;
        right = r;
    }
    public int numMatch (int x) {

       [2]

    }
}

class TreeLeaf extends IntStorage implements Countable {
    public TreeLeaf (int v) {
        value = v;
    }
    public int numMatch (int x) {
        if (x == value)
            return 1;
        else
            return 0;
    }
}
```

長崎大学大学院生産科学研究科 (博士前期課程)　　受験番号 _____
電気情報工学専攻 (情報システム工学系)
平成 20 年度 入学試験問題 ソフトウェア

解答欄

(2) 与えたプログラム中の空欄 [1], [2] を埋めて，任意の木に対して，引数で与えられた整数が含まれる回数を返す機能 numMatch() を完成させよ．以下は使用例である．

```
Countable root = new TriTree(new TriTree(new TreeLeaf(3),
                                         new TreeLeaf(1),
                                         new TreeLeaf(2)),
                             new BiTree(2,
                                        new TreeLeaf(2),
                                        new BiTree(1,
                                                   new TreeLeaf(3),
                                                   new TreeLeaf(4))),
                             new BiTree(2,
                                        new TreeLeaf(1),
                                        new TriTree(new TreeLeaf(1),
                                                    new TreeLeaf(2),
                                                    new TreeLeaf(3))));
System.out.println(root.numMatch(1));  // 4 を出力する
System.out.println(root.numMatch(2));  // 5 を出力する
System.out.println(root.numMatch(3));  // 3 を出力する
```

解答欄

[1]

[2]

4. 方程式 $f(x) = 0$ の解が区間 $a \leq x \leq b$ に存在するとき，解を求める方法として次のアルゴリズムを考える．ただし，関数 $f(x)$ は区間 $a \leq x \leq b$ で連続かつ単調とする．

1) 区間両端の関数値 $f(a), f(b)$ が異符号またはどちらかが 0 であれば，$f(x) = 0$ となる解が区間 $a \leq x \leq b$ のどこかに存在する．

2) 関数上の点 $(a, f(a))$ と $(b, f(b))$ を結ぶ直線を考え，その直線が x 軸と交わる点の x 座標を c とする．関数値 $f(c)$ を調べ，もし $f(a)$ と $f(c)$ が異符号またはどちらかが 0 であれば，解は区間 $a \leq x \leq c$ に存在する．そうでなければ，解は区間 $c \leq x \leq b$ に存在する．

3) 解の存在する区間について上記と同様の分割を繰り返し，解の存在区間をせばめていく．

4) 解の存在区間が十分狭くなったら，その区間から求められた c を $f(x) = 0$ の解とする．

区間 $2 \leq x \leq 3$ において $x^2 = 5$ の解を得るために，次のようなプログラムを作成した．空欄に適切なコードを挿入してプログラムを完成させよ．解は小数点以下 3 桁程度まで正しく求めるものとする．

```
#include <stdio.h>

double func(double x);
double calcc(double a, double b);

int main(void)
{
    double a=2.0, b=3.0, c;

    while( [ア] ){

        c = [イ]

        if( func(a)*func(c) <= 0 )
            [ウ]

        else
            [エ]

    }
    printf(" x=%lf\n", calcc(a,b) );

    return 0;
}

double func(double x)
{
    return( x*x-5 );
}

double calcc(double a, double b)
{
    [オ]

}
```

解答欄：

ア	
イ	
ウ	
エ	
オ	

5. n 個の整数が小さい順に配列 a[0], ···, a[n-1] に格納されているとき，新たな整数 x を，前後との大小関係が保たれるように配列 a に挿入する関数 ins( ) が下記のように与えられている．この関数を用いて，単純挿入法により整数をソートするプログラムを作成したい．以下の問いに答えよ．

```
int ins( int a[], int n, int x )
{
    int i;

    for(i=n-1; i>=0 && x<a[i]; i--)
    {
        a[i+1]=a[i];
    }
    a[i+1] = x;

    return 0;
}
```

(1) 整数 (40, 20, 50, 30, 10) を昇順に並べる場合を例として，単純挿入法によりソートを行う手順を具体的に説明せよ．手続きを時間順に箇条書きに記述して説明すること．

(2) 単純挿入法によるソートは「安定なソート」か「不安定なソート」かを述べよ．

(3) 以下の空欄に適切なコードを挿入して単純挿入法によりソートを行うプログラムを完成させよ．上記の関数 ins( ) が利用できるとして良い．

```
#include <stdio.h>
int ins( int a[], int n, int x );

int main(void)
{
    int a[5] = { 40, 20, 50, 30, 10 };
    int n=5, i;

    [                    ]

    for( i=0; i<n; i++)
        printf("%d ", a[i]);
    printf("\n");

    return 0;
}
```

解答欄：

(1)		(2)	
		(3)	