1. 下表はプロセスの到着時刻と必要計算時間を表わしたものである (単位は秒とする). このようにプロセスが生成されたとして, SJF スケジューリングアルゴリズムを用いた場合の平均ターンアラウンド時間を小数点以下 1 位まで求めよ. なお OS 内での処理に要する時間は無視してよい. またいずれのプロセスも入出力は行なわない. すなわち計算時間は実行時間に等しいものとする.

pid	到着時刻	必要計算時間
プロセス 1	0	3
プロセス 2	0	2
プロセス 3	1	4
プロセス 4	5	2

解答欄

2. プログラミング言語における「継承」または「関数型言語」のいずれかを選び, それがどのようなものかを例を使いながら 100 字以上で説明せよ.

解答欄

3. int 型の正整数 n（< 40）を引数に取り，以下の int 型整数を返す 1 引数関数 f() について以下の設問に答えよ．記述には C 言語を用いる．なおこの返値はフィボナッチ数列になっている．

引数として与えた正整数	返値
1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21
9	34
10	55
...	...

(1) 関数 f() のプロトタイプ宣言を示せ．

解答欄

(2) 空欄（ア ～ ウ）を埋めて関数 f() を定義せよ．ただし for 文や while 文などの繰り返し構文は使わないものとし，繰り返しが必要な場合は関数 f() に関する再帰呼び出しで実現すること．引数の値が妥当な範囲に入っていない場合について考慮する必要はない．解答は直接空欄に書き込むこと．

```
[ア          ] f( [イ          ] n)
{
   if (n < 3) {
      return 1;
  }
   else {
    [ウ                          ]
  }
}
```

(3) 前問の関数 f() を以下のように 4 引数の下請け関数 fac() を呼び出す関数 fa() に書き直し，関数 fac() は自分自身を再帰呼出しするが，fac の呼出しの記述は一カ所しかない関数として定義することで，より高速に計算させることができる．空欄（エ ～ カ）を埋めて，この考え方を使った関数 fa() および fac() の定義を完成させよ．ただし for 文や while 文などの繰り返し構文は使わないものとする．解答は直接空欄に書き込むこと．

```
[エ]  fa( [オ]  n)     /* エ，オは前問のア，イと同じ */
{
  if (n < 3) {
    return 1;
  }
  else {
    return fac(3, n, 1, 1);          /* fa は fac を 1 回しか呼び出さない */
  }
}

int fac (int i, int owari, int x_1, int x_2)
{
   if (i == owari) {
     return (x_1 + x_2);
   }
   else {
     カ
   }
}
```

4. 1 次元配列 a[ ] に N 個の実数値が格納されているとする．変数 x としてある数値を与えたとき，配列 a[ ] の要素の中から変数 x の値に近い順に M 個抽出したい．以下の設問に答えよ．

(1) まず $M = 1$ としたときを考える．変数 x の値に最も近い配列 a[ ] の要素 (変数 x との差の 2 乗が最小となる要素) を抽出して表示するプログラムを，以下の空欄 (ア，イ) を埋めて作成せよ．ただし，2 つの実数を引数として与えたとき，差の 2 乗を返す関数を diff( ) とする．新たな変数が必要であれば，適宜宣言して使用して良い．

```
#include <stdio.h>

#define N 10

double diff( double p, double q );

int main(void)
{
    double a[N] = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9, 11.0};
    double x, minval;
    int i, mi;

    printf("Input x:");
    scanf("%lf", &x);

    minval = diff( a[0], x );
    mi = 0;

    for(i=1; i<N; i++) {

        [ ア ]

    }

    /* x の値に最も近い a[ ] の要素を表示 */
    printf("添字: %d\n",mi);
    printf("値: %f\n",a[mi]);

    return 0;
}

double diff( double p, double q )
{

    [ イ ]

}
```

解答欄

ア	
イ	

(2) つぎに，$M \geq 2$ の一般の場合にも適用できるようにする．つまり，以下の手順により，配列 a[ ] の要素 a[0], ···, a[N-1] の中から，与えられた変数 x の値に近い順に M 個抽出して出力するプログラムを作成したい．ただし，$M \leq N$ とする．

i) 初期値として，a[0], ···, a[M-1] の値を，変数 x に近い順に別の配列 b[0], ···, b[M-1] に格納しておく．

ii) a[M], ···, a[N-1] の値を順番に調べ，b[M-1] よりも x に近い要素 a[i] が見つかるたびに以下を実行する．
- b[M-1] の値を，a[i] の値に変更する．
- b[0], ···, b[M-1] が x に近い順に保たれるように要素を並べ替える．

iii) 配列 b[0], ···, b[M-1] の値を画面に表示する．

前問で作成した関数 diff( ) が利用できるとして，以下の空欄 (ウ，エ) を埋めてプログラムを作成せよ．ただし，配列 b[ ] の要素を x に近い順に並べ替える関数を sortb( ) とする．新たな変数が必要であれば，適宜宣言して使用して良い．

```
#include <stdio.h>
#define N 10
#define M 5

double diff( double p, double q );
void sortb( double b[M], double x );

int main(void)
{
    double a[N] = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9, 11.0};
    double b[M];
    double x;
    int i;

    printf("Input x:");
    scanf("%lf", &x);


    for(i=0; i<M; i++) {
        b[i] = a[i];
    }
    sortb( b, x );

    for(i=M; i<N; i++) {

            ウ

    }

    /* x の値に近い順に配列の要素を表示 */
    for(i=0; i<M; i++) {
        printf("%d: %f\n", i, b[i] );
    }

    return 0;
}
```

(次ページへ続く)

解 答 欄

ウ	

```
void sortb( double b[M], double x )
{
    エ
}

double diff( double p, double q )
{
    省略
}
```

解答欄

エ	