長崎大学大学院生産科学研究科 (博士前期課程)
電気情報工学専攻 (情報システム工学系)
平成 21 年度入学試験問題 ソフトウェア　　受験番号

1. 以下の 2 つの関数はそれぞれ別の手法で等価な計算を行なおうとしたものである．記述言語は C 言語である．これに関し，次の設問に答えよ．解答は次ページ以降の解答用紙にすること．

```
int f_l (int *vec, int size)
{
   int pr = 1, *ptr;
   for (ptr = vec; ptr < vec + size; ptr++) {
      pr *= *ptr;
   }
   return pr;
}

int f_r (int *vec, int size)
{
   if (size <= 0) return 1;
   return *vec * f_r(_______, size-1);
}
```

(1) 二つの関数が同じ計算をするように関数 `f_r` 中の下線部を埋めよ．

(2) 上記の関数を以下の関数 `vec2double` から呼ぶ．コンパイル時に型エラーにならないように 3 つの下線部を適切に埋めよ．解答欄にはこの関数の定義そのものを記述すること．

```
/* 与えられた配列（とその大きさ）から f の計算結果を double で返す */
double vec2double (_______ array, _______ size)
{
  return _______ f_l(array, *size); /* 等価であるので f_r を呼び出しても同じ */
}
```

(3) 関数 `f_l` 中の 2 箇所の下線部のみを変更して，配列中の最小値を返す関数 `f_min` を定義せよ．

(4) 0 を要素として含む配列が引数として与えられる確率が高いことがわかった．この事実を考慮して `f_l` を書き直し（以降この関数を `f0_l` と呼ぶ），コンパイラによる最適化が期待できない場合でも，実行時間の削減を実現せよ．解答欄には `f0_l` の定義を記述すること．なお，`f0_l` からは元の関数 `f_l`, `f_r` を呼ばないこと．

(5) コンパイラによって行なわれる，ループに関する代表的な最適化手法を 2 つ説明せよ．

(6) コンパイラによる最適化が期待できなければ，多くの場合において `f_r` は `f_l` よりも遅くなる．両者のメモリの使用状況に関する違いに注目し，その理由を説明せよ．

(7) 関数 `f_r` は末尾再帰 (tail recursion) の形をしていないため，単なる末尾再帰最適化では `f_l` と同じ計算量を必要とする機械語列には変換できない．引数の型や個数なども変更してよいものとして，`f_r` を基に，末尾再帰をする（従って最適化コンパイラによって `f_l` と同じ時間・空間計算量が期待できる）`f_tr` を定義せよ．なお引数の定義を変更した場合はこの関数をどのように呼び出せばよいか記述すること．

(8) Ocaml, Haskell などの言語が属す関数型プログラミングパラダイムの特徴の一つは関数が「純関数 (pure function)」であることである．ここで定義した関数 `f_r`, `f_tr` に関してどの主張が正しいか，以下の選択肢から一つ選べ．

(a) `f_r` は純関数であるが，`f_tr` は純関数でない．

(b) `f_r` は純関数でないが，`f_tr` は純関数である．

(c) `f_r` も `f_tr` も純関数である．

(d) `f_r` も `f_tr` も純関数でない．

解答用紙 1 ページ目 (解答欄は自由に作成してよい．2 ページで足りない場合は裏面を使用すること．)

解答用紙 2 ページ目

2. N 個の文字列が，下図の例のように N 行 M 列の 2 次元配列 dat[ ][ ] に格納されている．それぞれの文字列の直後には，文字列の終端を示す NULL コード (\0) が置かれている．これらの文字列の中に指定したものがあるかどうかを検索したい．

以下の設問においては，必要な変数は適宜宣言して使用して良い．ただし，C 言語の標準ライブラリとして用意されている文字列関数（strcpy( ) や strcmp( ) など）は使用しないものとする．

dat[i][j]

i=0	a	p	p	l	e	\0			
1	o	r	a	n	g	e	\0		
2	b	a	n	a	n	a	\0		
3									

(1) 文字列を前から順番に比較する方法 (逐次探索) によって，指定した文字列が存在するかどうかを調べ，見つかった時はその位置 (2 次元配列 dat[i][j] の添字 i の値) を返し，見つからなかった時は -1 を返す関数 ssearch( ) を，以下の空欄を埋めて作成せよ．

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

int ssearch( char a[][M], char b[] );

int main(void)
{
    char dat[N][M] = {"apple","orange","banana","peach","grape"};
    char str[M] = "grape";
    int pos;

    pos = ssearch( dat, str );
    if( pos >= 0 )
        printf("found at pos=%d\n", pos );
    else
        printf("not found\n");
    return 0;
}

int ssearch( char a[][M], char b[] )
{
    int i,j;

    for(i=0; i<N; i++) {

        ア

    }
    return -1;
}
```

解 答 欄

ア	

(2) 逐次探索では後ろにある項目ほど探索に時間がかかるので，検索の頻度が高い項目をなるべく前に配置しておくと検索時間が短くなる．配列の $p+1$ 番目 (添字: p) の文字列を，配列の先頭に移動し，1 番目から p 番目までの文字列をひとつずつ後ろへ移動させる関数 sshift( ) を，以下の空欄を埋めて作成せよ．

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

void sshift( char a[][M], int p );

int main(void)
{
    char dat[N][M] = {"apple","orange","banana","peach","grape"};
    int i, pos = 4;

    for( i=0; i<N; i++)
        printf("%s ", dat[i] );
    printf("\n");

    sshift( dat, pos );

    for( i=0; i<N; i++)
        printf("%s ", dat[i] );
    printf("\n");

    return 0;
}

void sshift( char a[][M], int p )
{
    int i,j;
    char sbuf[M];

    if( p == 0 )
        return;

    for( j=0; a[p][j]!='\0'; j++ )
        sbuf[j] = a[p][j];
    sbuf[j] ='\0';

    [ イ ]

}
```

解 答 欄

イ	

(3) 検索によって見つかった項目を，検索のたびに配列の先頭に移動させることによって，検索の頻度が高い項目を前のほうに配置したい．以下の空欄を埋めて，(1) で作成した関数 ssearch( ) を変更して新しい関数 sm_search() を作成せよ．ただし，(2) で作成した関数 sshift( ) が利用できるものとしてよい．

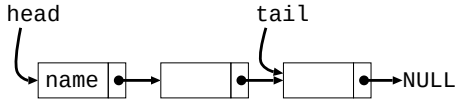
```
#include <stdio.h>
#define N 5
#define M 10

int sm_search( char a[][M], char b[] );
int sshift( char a[][M], int p );

int main(void)
{
    char dat[N][M] = {"apple","orange","banana","peach","grape"};
    char str[M] = "orange";
    int pos;

    pos = sm_search( dat, str );
    if( pos >= 0 )
        printf("found at pos=%d\n", pos );
    else
        printf("not found\n");

    return 0;
}

int sm_search( char a[][M], char b[] )
{

    [ ウ ]

}

int sshift( char a[][M], int p )
{

    省略

}
```

解 答 欄

ウ	

3. キーボードから N 人の名前を小文字のアルファベット (30 文字以下) で入力し，構造体をノードとする下図のようなリスト構造に格納したい．head と tail は，それぞれリストの先頭と末尾を指すポインタとする．

以下の設問においては，必要な変数は適宜宣言して使用して良い．ただし，C 言語の標準ライブラリとして用意されている文字列関数は使用しないものとする．



(1) 入力した名前を入力した順に連結してリスト構造を作成するプログラムを以下の空欄を埋めて作成せよ．

```
#include <stdio.h>
#include <stdlib.h>
#define N 10

struct NODE
{
    char         name[31];
    struct NODE  *next;
};

int main(void)
{
    struct NODE *head, *tail, *trace, *work;
    int i;

    work = malloc(sizeof *work);
    head = tail = work;
    printf("Input name:");
    scanf("%s", work->name);
    work->next = NULL;

    for(i=1; i<N; i++) {
        work = malloc(sizeof *work);
        printf("Input name:");
        scanf("%s", work->name);

            ア

    }

    for(trace = head; trace != NULL; trace = trace -> next) {
        printf("%s  ", trace->name);
    }
    printf("\n");

    return 0;
}
```

解 答 欄

ア	

(2) 入力した名前がアルファベット順 (辞書順) に並ぶようにリスト構造を作成したい．次の設問で使用するために，任意の文字列 s と t を引数として与えたときどちらが先になるかを判定する関数 compare( ) を，以下の空欄を埋めて作成せよ．ただし，関数 compare( ) は，文字列 s が先になるとき 1 を返し，文字列 t が先になるとき −1 を返し，両者が同一であるとき 0 を返すものとする．

```
int compare( char *s, char *t )
{
    while( 1 ){

        [ イ ]

        s++;
        t++;
    }
}
```

解 答 欄

イ	

(3) (2) で作成した関数 compare( ) が利用できるものとして，名前が新たに入力されると，アルファベット順の適切な位置にその文字列を挿入してリスト構造を作成するプログラムを，以下の空欄を埋めて作成せよ．ただし，ここではリストの末尾を指すポインタ (tail) は用いないものとする．

```
#include <stdio.h>
#include <stdlib.h>
#define N 10
int compare( char *s, char *t );

struct NODE
{
    char         name[30];
    struct NODE  *next;
};

int main(void)
{
    struct NODE *head, *trace, *work;
    int i;

    work = malloc(sizeof *work);
    head = work;
    printf("Input name:");
    scanf("%s", work->name);
    work->next = NULL;

    for(i=1; i<N; i++) {
        work = malloc(sizeof *work);
        printf("Input name:");
        scanf("%s", work->name);

        if( compare( head->name, work->name) <= 0 ){

            ウ

        }
        else{
            for(trace=head; trace->next!=NULL; trace=trace->next) {

                エ

            }
            if(trace->next==NULL){

                オ

            }
        }
    }
    return 0;
}
```

解答欄：

ウ	
エ	
オ	