アルゴリズムとデータ構造　第６回演習　　年　　組　　番 氏名

1. 下記に示すのは，**動的に確保したメモリ**に double 型の数値を n 個読み込み，それらを小さい順に画面に表示するプログラムである。空欄を適切に埋めてプログラムを完成させよ。並べ替えるための関数は第 3 回または第 4 回に説明したものを使うようにして，関数の本体はここに書かなくてよい。

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    double *data;
    int i, n;
    int nbytes;

    printf("n? ");
    scanf("%d", &n);

    nbytes =

    data = (double *) malloc(nbytes);

    for (i = 0; i < n; i++) {
        printf("data[%d]? ", i);

        scanf("%lf",         );
    }



    printf("¥n--¥n");
    for (i = 0; i < n; i++) {
        printf("%f¥n", data[i]);
    }
    free(data);
    return 0;
}
```

2. 下記は商品コード（code）と価格（price）を持つ構造体 struct item である。この**構造体の配列**の要素を，価格が**高い順**にクイックソートで整列する関数 sort_by_price を作成したい。空欄を適切に埋めよ。

```
struct item {
    int code;
    int price;
};

void sort_by_price(struct item a[], int left, int right)
{
   int l, r;
   struct item pivot, t;

   if (left >= right) return;

   pivot =

   l = left;  r = right;
   while (1) {
       while (                  ) {

       }
       while (                  ) {

       }
       if (l >= r) break;



       l++; r--;
   }
   sort_by_price(a,         ,         );

   sort_by_price(a,         ,         );

}
```

3. 下記は，最も単純な**文字列検索**のアルゴリズムである。この関数は，文字列 text の中に文字列 word が含まれるか先頭から調べて最初に見つかった位置（ポインタ）を返す。ただし，見つからなかった場合には NULL ポインタを返す。空欄を適切に埋めて動作を確認せよ。

```
#include <stdio.h>
#include <string.h>

char *strfind(char *text, char *word)
{
    int tlen, wlen, t, w;

    tlen = strlen(text);
    wlen = strlen(word);

    for (t = 0; t <=          ; t++) {

        for (w = 0; w <       ; w++) {

            if (text[      ] != word[    ])
                break;
        }
        if (w == wlen) return text + t;
    }

    return
}
```

```
int main(void)
{
    char str1[1024], str2[256];
    char *pos;

    scanf("%[^¥n]", str1);
    scanf(" ");
    scanf("%s", str2);

    pos = strfind(str1, str2);

    if (pos != NUll)
        printf("Found at %ld.¥n",
            pos - str1);
    else
        printf("Not found.¥n");
    return 0;
}
```

4. 下記は，**ポインタの配列**を理解するための例である。構造体 struct card はトランプのカードを表すが，画像などの付加情報によってサイズが大きいものと想定する。そこで，直接構造体の配列を作らず，構造体のポインタの配列 deck を作っている。この要素はポインタなので，使用にあたっては malloc で実際のメモリ領域を割り当てなければならない。空欄を埋めて動作を確認し，この方法の利点を考えよ。

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

struct card {
    int suit;  /* 0〜3 */
    int rank;  /* 1〜13 */
    /* ここに大量の付加情報を想定 */
};

/* Card を struct card の別名とする */
typedef struct card Card;

/* 各要素がポインタである配列 */
Card *deck[52];

int main(void)
{
    int i, j, k;
    Card *ptr;
    Card *tmp;

    /* 乱数の準備 */
    srand(time(NULL));
```

```
    /* カードの生成 */
    for (i = 0; i < 52; i++) {

        ptr = (Card *) malloc(                );
        ptr->suit = i % 4;


        deck[i] = ptr;
    }

    /* シャッフル*/
    for (k = 0; k < 100; k++) {
        i = rand() % 52;
        j = rand() % 52;
        tmp = deck[i];
        deck[i] = deck[j];
        deck[j] = tmp;
    }

    /* 結果表示 */
    for (i = 0; i < 52; i++) {



    }
}
```