平成 25 年 8 月 3 日（土）9:00～12:00

大阪大学大学院情報科学研究科

コンピュータサイエンス専攻
情報システム工学専攻
情報ネットワーク学専攻
マルチメディア工学専攻
バイオ情報工学専攻

平成 26 年度　博士前期課程　入試問題

（A）情報工学

【注意事項】

- 問題数は必須問題 2 題（問題 1～2），選択問題 5 題（問題 3～7），合計 7 題である．
  必須問題は 2 題すべて解答すること．また，選択問題は 2 題を選択して解答すること．
- 問題用紙は表紙を含めて 13 枚である．
- 解答用紙は全部で 5 枚ある．
  1 枚目（赤色）の解答用紙には問題 1（必須問題）の解答を
  2 枚目（緑色）の解答用紙には問題 2（必須問題）の(1)の解答を
  3 枚目（緑色）の解答用紙には問題 2（必須問題）の(2)の解答を
  4 枚目（白色）の解答用紙には問題 3～7（選択問題）から選択した 1 題の解答を
  5 枚目（白色）の解答用紙には問題 3～7（選択問題）から選択したもう 1 題の解答を
  それぞれ記入すること．
  解答用紙は間違えると採点されないことがあるので注意すること．
- 解答用紙は 5 枚すべてを回収するので，すべての解答用紙に受験番号を記入すること．
- 解答用紙の「試験科目」の欄には解答した問題の科目名（「アルゴリズムとプログラミング」など）を記入すること．
  また，選択問題調査票には，選択した問題の番号（3～7 から二つ）に○をつけること．
- 解答欄が不足した場合は裏面を使用すること．その際，表面末尾に「裏面に続く」と明記しておくこと．解答用紙の追加は認めない．
- 解答用紙には，日本語または英語で解答すること．

1 【必須問題】アルゴリズムとプログラミング

配点：(1) 30 点，(2) 20 点，(3) 15 点，(4) 25 点，(5) 20 点，(6) 15 点

図 1 は ANSI-C 準拠である C 言語のプログラム（program）である．このプログラムにおいて，insert 関数はある特定の規則に従ってデータ列（data sequence）に新しいデータ（data）を挿入（insert）する関数（function）であり，データ列を格納する配列（array）A，配列の大きさ SIZE，挿入するデータ d を引数（arguments）とする．delete 関数はデータ列からある特定の規則に従ってデータを一つ取り出し（retrieve），削除（delete）する関数であり，データ列を格納する配列 A を引数とし，取り出したデータを戻り値（return value）とする．main 関数は，それらの二つの関数を実行する一例を示している．以下の各問に答えよ．

(1) 図 1 のプログラムにおいて，43 行目および 46 行目の処理を実行した結果をそれぞれ示せ．

(2) 図 1 のプログラムにおいて，43 行目の処理を実行した直後の変数 front および変数 rear の値をそれぞれ示せ．

(3) 図 1 の insert 関数内の 11～21 行目では，新たに挿入するデータの挿入位置を決定している．この際，データ列を最初から一つずつ調べることなく，処理を効率化している．具体的にどのようなことをしているかを簡潔に説明せよ．

(4) データ列のデータ数を n としたとき，insert 関数および delete 関数を実行する際の時間計算量（time complexity）のオーダー（order）を示せ．その理由も簡潔に説明せよ．

(5) 図 1 の main 関数において，47 行目以降に insert 関数および delete 関数を多数回実行した場合，データ列のデータ数に関わらず，実行途中でデータの挿入に失敗してしまう．その原因，および，データの挿入が失敗する条件を，データ挿入回数の観点から簡潔に説明せよ．

(6) 上記(5)の問題を解決するために，配列を十分大きくする，および，データの挿入回数に制限を設ける以外に，どのような方法があるか，簡潔に説明せよ．ただし，その方法が insert 関数および delete 関数の時間計算量のオーダーを変えることがあってはならない．また，できるだけ時間計算量の増加が少ない方法を示すこと．なお，その方法を実現するために，関数に新たな引数を追加することがあってもよいが，新たな関数は追加してはならない．

```
1  #include <stdlib.h>
2  #include <stdio.h>
3
4  int front = 0;
5  int rear = 0;
6
7  void insert(int A[], int SIZE, int d) {
8    int p, left, right, m, i;
9    if (rear > SIZE-1) { printf ("Overflow !!\n"); exit(1); }
10
11   p = -1;
12   if (front == rear) p = front;
13   else {
14     left = front; right = rear-1;
15     while (left < right) {
16       m = (left+right)/2;
17       if (A[m] == d) { p = m+1; break; }
18       if (d < A[m]) right = m-1; else left = m+1;
19     }
20     if (p == -1) { if (A[left] > d) p = left; else p = left+1; }
21   }
22
23   i = rear;
24   while (i > p) { A[i] = A[i-1]; i--; }
25   A[p] = d; rear++;
26 }
27
28 int delete(int A[]) {
29   int x;
30   if (front == rear) { printf ("Underflow !!\n"); exit(1); }
31   x = A[front]; front++;
32   return x;
33 }
34
35 int main () {
36   int i;
37   int A[20]; int SIZE = 20;
38
39   int a1[5] = {10, 5, 20, 6, 13};
40   int a2[5] = {2, 5, 18, 7, 5};
41
42   for (i = 0; i < 5; i++) insert(A, SIZE, a1[i]);
43   for (i = 0; i < 3; i++) printf("%d ", delete(A)); printf ("\n");
44
45   for (i = 0; i < 5; i++) insert(A, SIZE, a2[i]);
46   for (i = 0; i < 7; i++) printf("%d ", delete(A)); printf ("\n");
47
48   return 0;
49 }
```

図 1：プログラム

left

right 存序

0 0 0 0 0

1 25 50 75 100.

mid = left + right+1 / 2.

v1 < value. right = mid - 1

v1 >= value left = mid.

3种 <

= break

>

二分探索法

1. 5 6 10
   2 5 5 7 13 18 20

2. front = 3
   rear = 5

3. Bは常に昇順にソートされている。
   二分探索の計算時間はO(logn)

4. insert = O(n)
   delete = O(1)

5. 2回目以降

6. frontやrearが配列の末端に到達したら
   先頭に移動するようにしてループ構造にする
   またはinsertの実行時にデータ列を後ろでなく前に
   詰めるようにする。