FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110211 Introductions to Data Structure

YEAR II, Summer Semester, Midterm Examination, April 29, 2010, Time 09:00 – 12:00

ชื่อ-นามสกุล_____เลขประจำตัว | | | | | | | | | 2 | 1 | CR58_____

หมายเหตุ

1. ข้อสอบมีทั้งหมด 7 ข้อในกระดาษคำถามคำตอบจำนวน 5 แผ่น 5 หน้า คะแนนเต็ม 70 คะแนน
2. ไม่อนุญาตให้นำตำราและเครื่องคำนวณต่างๆ ใดๆ เข้าห้องสอบ
3. ควรเขียนตอบด้วยลายมือที่อ่านง่ายและชัดเจน สามารถใช้ดินสอเขียนคำตอบได้
4. ห้ามการหยิบยืมสิ่งใดๆ ทั้งสิ้น จากผู้สอบอื่นๆ เว้นแต่ผู้คุมสอบจะหยิบยืมให้
5. ห้ามนำส่วนใดส่วนหนึ่งของข้อสอบออกจากห้องสอบ ข้อสอบเป็นทรัพย์สินของราชการซึ่งผู้ลักพาอาจมีโทษทางคดีอาญา
6. ผู้ที่ประสงค์จะออกจากห้องสอบก่อนหมดเวลาสอบ แต่ต้องไม่น้อยกว่า 45 นาที
7. เมื่อหมดเวลาสอบ ผู้เข้าสอบต้องหยุดการเขียนใดๆ ทั้งสิ้น
8. ผู้ที่ปฏิบัติเข้าข่ายทุจริตในการสอบ ตามประกาศคณะวิศวกรรมศาสตร์
   มีโทษ คือ ได้รับ สัญลักษณ์ F ในรายวิชาที่ทุจริต และพักการศึกษาอย่างน้อย 1 ภาคการศึกษา

รับทราบ

ลงชื่อนิสิต (……..……)

หมายเหตุ (เพิ่มเติม)

1. สำหรับข้อที่ให้ออกแบบ หรือ เขียนโปรแกรม คะแนนที่ได้จะแปรตามประสิทธิภาพในการทำงานของโปรแกรม
2. สำหรับข้อที่ให้วิเคราะห์เวลาการทำงาน คะแนนที่ได้จะแปรตามความใกล้เคียงความเป็นจริงของการวิเคราะห์
3. **นิสิตสามารถอ้างถึงและเรียกใช้คลาสต่าง ๆ ที่อยู่ในเอกสารประกอบการสอนได้โดยไม่จำเป็นต้องเขียนขึ้นมาใหม่**
4. **ในข้อที่ต้องออกแบบโครงสร้างข้อมูล นิสิตไม่จำเป็นต้องตรวจสอบถึงกรณีที่มีการใส่ข้อมูลเข้าไปมากกว่าเนื้อที่ที่มีอยู่ (เสมือนว่าการจองพื้นที่นั้นจองมากพอเสมอ) หรือ กรณีที่เอาข้อมูลออกเมื่อไม่มีข้อมูลอยู่ในโครงสร้างข้อมูล**
5. **ให้เขียนคำตอบลงในเฉพาะพื้นที่ที่เว้นว่างไว้**
6. ให้นิสิตเขียนรหัสประจำตัวและเลขที่ใน CR58 ในทุกหน้าของกระดาษคำถามด้วย
7. สามารถเขียนคำตอบด้วยภาษา C++ หรือ Java ก็ได้ แต่ในกรณีที่เขียนด้วย Java นิสิตจะต้องแปลงโจทย์ให้เป็นภาษา Java ด้วยตัวเอง พร้อมทั้งเขียนโจทย์ที่แปลงลงในคำตอบด้วย

---

1. (5 คะแนน) กำหนดให้มีตัวแปร set เป็นประเภท ArraySet<int> ที่มีขนาดใหญ่เพียงพอที่จะเก็บข้อมูล จงแสดงผลลัพธ์ที่เกิดขึ้นบนหน้าจอของส่วนของโปรแกรมต่อไปนี้

```
ArraySet<int> *set = new ArraySet<int>( b );
for (int i = 1;i <= a;i++)
    if (a % i == 0)
        set->add(i);   set->add(a / i);
for (int i = b;i > 0;i--)
    if (b % i == 0 && set->contains(i))
        printf("%d \n",i); exit(0)
```

……
……
……

2. (5 คะแนน) กำหนดให้มีตัวแปรดังต่อไปนี้ bh เป็น BinaryHeap<int> q เป็น ArrayQueue<int> และ s เป็น ArrayStack<int> และให้ตัวแปรทั้งหมดมีขนาดใหญ่เพียงพอที่จะเก็บข้อมูล จากส่วนของโปรแกรมต่อไปนี้ จงระบุผลลัพธ์ที่เกิดขึ้นบนหน้าจอ

```
bh.enqueue(3); bh.enqueue(1); bh.enqueue(2); bh.enqueue(4);
q.enqueue(bh.dequeue());    q.enqueue(bh.dequeue());
q.enqueue(bh.dequeue());
s.push(q.dequeue()); s.push(q.dequeue());
s.push(q.dequeue());
printf("%d \n",s.pop()); printf("%d \n",s.pop());
printf("%d \n",s.pop());
```

……
……
……

3. (10 คะแนน) จงเขียนฟังก์ชัน int findMaxDiff() สำหรับโครงสร้างข้อมูล ArrayList<int> เพื่อทำการหาค่าผลต่างที่มากที่สุดของข้อมูลสองตัวใด ๆ ใน ArrayList<int> ดังกล่าว

```
template <class T>
class ArrayList : public List<T> {
    protected:
        T *elementData;
        int mySize;
    public: // ไม่ได้แสดงฟังก์ชันต่าง ๆ ของ ArrayList แต่สามารถเรียกใช้งานได้ตามปรกติ
        int findMaxDiff() {









        }
}
```

4. (15 คะแนน) จงออกแบบโครงสร้างข้อมูล StackByQueue ซึ่งเป็นโครงสร้างข้อมูลประเภท Stack โดยที่โครงสร้างข้อมูลดังกล่าวจะใช้โครงสร้างข้อมูลแบบ ArrayQueue ในการเก็บข้อมูล ให้นิสิตเขียนฟังก์ชัน T pop() และ T peek() ให้ถูกต้อง **พร้อมทั้งระบุเวลาในการทำงานของฟังก์ชันดังกล่าว (ไม่ต้องแสดงวิธีทำ)** โดยมีการกำหนดส่วนของโปรแกรมบางส่วนมาให้แล้วดังต่อไปนี้

```
template <class T>
class StackByQueue : public Stack<T> {
    private:
        ArrayQueue<T> *q;  int cap;
    public:
        StackByQueue(int cap) {
            q = new ArrayQueue<T>(cap); this->cap = cap;
        }

        void push(T e) { q->enqueue(e);  }
        void isEmpty() { return q->isEmpty(); }
        void size() {return q->size(); }

        T peek() {









        }

        T pop() {









        }
};
```

วิเคราะห์เวลาการทำงาน

……
……
……

5. (10 คะแนน) กำหนดให้มีฟังก์ชัน `void multiDequeue(int num)` เพิ่มขึ้นมาสำหรับโครงสร้างข้อมูลประเภท `ArrayQueue` ซึ่งจะมีการทำงานเป็นการ `dequeue` ข้อมูลใน `Queue` ทิ้งไปหลาย ๆ ตัว และให้มีฟังก์ชัน `main` ที่เรียกใช้งานฟังก์ชันดังกล่าว โดยมีรายละเอียดดังส่วนของโปรแกรมด้านล่างนี้ **จงระบุเวลาที่ใช้ในการทำงานของฟังก์ชัน muliDequeu และ main พร้อมทั้งยกเหตุผลประกอบ**

```
template <class T>
class ArrayQueue : public Queue<T> {
    public: // ไม่ได้แสดงฟังก์ชันต่าง ๆ ของ ArrayQueue แต่สามารถเรียกใช้งานได้ตามปรกติ
      void multiDequeue(int num) {
          for (int i = 0;i < num;i++)
              dequeue();
      }
};

int main(int argc,char **argv) {
    int n; scanf("%d\n",&n);
    ArrayQueue<int> *q = new ArrayQueue(n);

    for (int i = 0;i <n;i++)
        for (int j = 0;j < n;j++)
          q->enqueue(i);

    for (int j = 0;j < n;j++)
        q->multiDequeue(1);
}
```

……
……
……
……
……
……
……
……
……
……

6. (15 คะแนน) จงปรับปรุง `class ArrayCollection` ดังต่อไปนี้

   a. ให้เขียนฟังก์ชัน int `getDistinct()` สำหรับโครงสร้างข้อมูล `ArrayCollection` เพื่อนับว่าข้อมูลใน `ArrayCollection` ดังกล่าว มีข้อมูลที่มีค่าแตกต่างกันทั้งหมดกี่ตัว (คำแนะนำ: ควรจะพิจารณาการนำเอาโครงสร้างข้อมูลอื่นเข้ามาช่วยหาคำตอบ)

   b. ให้เขียนฟังก์ชั่น `int getCount(T e)` เพื่อนับว่าใน ArrayCollection ดังกล่าว มีข้อมูล e เกิดขึ้นอยู่กี่ครั้ง ถ้าไม่มีข้อมูลเลยให้คืนค่า 0

ตัวอย่างเช่น ถ้า `ArrayCollection` มีข้อมูลเป็น 10,20,30,10 ฟังก์ชัน `getDistinct()` จะต้องมีค่าเป็น 3 เนื่องจากมีข้อมุลที่ต่างกันอยู่เพียงแค่ 3 ตัวคือ 10,20 และ 30 ในขณะที่ `getCount(10)` จะต้องมีค่าเป็น 2 และ `getCount(30)` มีค่าเป็น 1

```
template <class T>
class ArrayCollection : public Collection<T> {
    private:
        T *elementData;  int cap; int mySize;
    public: // ไม่ได้แสดงฟังก์ชันต่าง ๆ ของ ArrayCollection แต่สามารถเรียกใช้งานได้ตามปรกติ
        int getDistinct() {









        }
        int getCount(T e) {









        }
};
```

7. (10 คะแนน) สมมติให้ฟังก์ชัน `fixDown(int idx)` ของโครงสร้างข้อมูลประเภท `BinaryHeap` นั้นมีการเปลี่ยนแปลงไปเป็นดังส่วนของโปรแกรมด้านล่างนี้ (ให้สังเกตว่า `fixDown` ใหม่นี้ ไม่มีการเปรียบเทียบระหว่างลูกตัวที่มากที่สุดกับตัวพ่อ แต่จะมีการเรียกใช้ `fixUp` ในตำแหน่งท้ายสุดแทน) จงให้เหตุผลว่า ทำไม `fixDown` ใหม่นี้สามารถทำงานได้อย่างถูกต้อง และ **จงยกตัวอย่างข้อมูลใน `BinaryHeap` (โดยการวาดต้นไม้ `BinaryHeap`) ที่ `fixDown` ใหม่นี้ ใช้จำนวนครั้งในการเปรียบเทียบข้อมูลภายใน `BinaryHeap` โดยรวมในส่วน `fixUp` ด้วยแล้ว น้อยกว่า `fixDown` ตามแบบปรกติ**

```
void fixDown(int idx) {
    int c;
    while ( (c = idx * 2 + 1) < mySize) {
        if (c + 1 < mySize && elementData[c + 1] > elementData[c]) c++;
        swap(idx,c);
        idx = c;
    }
    fixUp(idx);
}
```

……
……
……
……
……
……
……