FACULTY OF ENGINEERING

CHULALONGKORN UNIVERSITY

2110211 Introduction to Data Structures

YEAR II, Second Semester, Midterm Examination, December 25, 2012, Time 13:00 – 16:00

---

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

หมายเหตุ

1. ข้อสอบมีทั้งหมด 9 ข้อในกระดาษคำถามคำตอบจำนวน 8 แผ่น 8 หน้า คะแนนเต็ม 101 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 (.....)

---

1. (6 คะแนน) จงวาด heap (แบบ min-heap) ที่ได้จากการค่อย ๆ เพิ่มข้อมูล 7, 15, 8, 4, 9, 5, 10, 11, 4 ตามลำดับ (วาดการเปลี่ยนแปลงของต้นไม้หลังการเพิ่มข้อมูลแต่ละตัว)

2. (16 คะแนน) จงอธิบายสั้น ๆ ว่า แต่ละฟังก์ชันข้างล่างนี้ทำอะไร และวิเคราะห์ประสิทธิภาพการทำงานเชิงเวลาด้วย (ให้ถือว่าการเปรียบเทียบข้อมูลด้วย cmp ใช้เวลาเป็น $\Theta(1)$ )

```
// ฟังก์ชันนี้เพิ่มอยู่ใน stack
void what01(Stack s) {
  int i, n = s->size;
  for (i=0; i<n; i++) {
    DType t = s->data[i];
    s->data[i] = s->data[n – i – 1];
    s->data[n – i – 1] = t;
  }
}
```

```
// ฟังก์ชันนี้เพิ่มอยู่ใน binaryHeap (ที่เป็น min Heap)
BHeap what02(binaryHeap h) {
  int n = h->size;
  if (n < 2) System.exit(-1);
  if (n == 2) return h->data[1];
  if (cmp(h->data[1], h->data[2]) < 0)
    return h->data[1];
  else
    return h->data[2];
}
```

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

```
// ฟังก์ชันนี้เพิ่มอยู่ใน binaryHeap (ที่เป็น min Heap)
int what03(DType d[], int n) {
  int c;
  for (c=n-1; c>=1; c--) {
    int p = (c-1)/2;
    if (cmp(d[c],d[p])>0) return 0;
  }
  return 1;
}
```

```
// ฟังก์ชันนี้เพิ่มอยู่ใน AList
int what04(AList list) {
  int i, j;
  for (i = list->size-1; i>=0; i--) {
    DType x = getAList(list,i);
    int j = indexOfAList(x);
    if (i != j) return 0;
  }
  return 1;
} // หมายเหตุ: indexOfAList ค้นจากซ้ายไปขวา
```

3. (5 คะแนน)

ฟังก์ชัน **infix2postfix** ทางขวานี้ หานิพจน์แบบ postfix จากนิพจน์แบบ infix ที่ได้รับ ฟังก์ชันนี้เหมือนกับที่นำเสนอในชั้นเรียน บรรทัดที่แตกต่างจากที่นำเสนอคือ สามบรรทัดแรก

กำหนดให้นิพจน์มี operator **+ - * / ^ % @ $** และ ( กับ ) โดย

**+ - * /** แทน บวก ลบ คูณ หาร

**^** แทนการยกกำลัง

**%** แทนการหาเศษของการหาร

**@** แทนการหารากเช่น **27 @ 3** คือ $\sqrt[3]{27}$ มีค่าเป็น 3

**$** แทนการหาร้อยละ เช่น **200 $ 10** แทน 10% ของ 200 คือ 20

กำหนดให้ลำดับการคำนวณของ **^** และ **@** เป็นจากขวามาซ้าย เช่น

**2^4^2 = 2^(4^2)** ซึ่งคือ $2^{4^2} = 2^{16}$,

**256@8@3 = 256@(8@3)** ซึ่งคือ $\sqrt[3]{8}\sqrt{256} = \sqrt[2]{256}$

ส่วนลำดับการคำนวณของ **$** เป็นจากซ้ายไปขวา เช่น

**200$25$10 = (200$25)$10** มีค่าเท่ากับ 5

นอกจากนี้ กำหนดให้ความสำคัญของคำนวณ **%** เท่ากับ การคูณ และหาร และให้ความสำคัญของ เท่ากับการยกกำลัง และให้ให้ความสำคัญของ **$** นั้นน้อยกว่า การบวกและการลบ

จงเติมค่าในอาเรย์ **outPriorities** และ **inPriorities** เพื่อให้ลำดับการคำนวณเป็นไปตามที่กำหนดไว้ข้างต้น

```
char operators[] = "+-*/^%@$()";
int outPriorities[] = {                    };
int  inPriorities[] = {                    };
int indexOf(char *ops, char c) {
  int i=0;
  while (ops[i] != '\0') {
    if (c == ops[i]) return i;
    i++;
  }
  return -1;
}
int inPriority(char op) {
  return inPriorities[indexOf(operators, op)];
}
int outPriority(char op) {
  return outPriorities[indexOf(operators,op)];
}
int isOperator(char c) {
  return indexOf(operators, c) >= 0;
}
void infix2postfix(char *infix,char *postfix){
  Stack s = newStack(10);
  int i, n, j = 0;
  for (i=0, n=strlen(infix); i<n; i++) {
    char token = infix[i];
    if (!isOperator(token)) {
      postfix[j++] = token;
    } else {
      int p = outPriority(token);
      while(!isEmptyStack(s) &&
            inPriority(top(s)) >= p) {
        postfix[j++] = pop(s);
      }
      if (token == ')')
        pop(s); // this must be (
      else
        push(s, token);
    }
  }
  while(!isEmptyStack(s)) postfix[j++]=pop(s);
  postfix[j] = '\0';
  freeStack(s);
}
```

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

4. (5 คะแนน) จงเติมคำสั่งในช่องว่างของฟังก์ชัน removeAny และ containsAny เพื่อให้ทำงานตามที่เขียนใน comment

```
#include <stdio.h>
#include <stdlib.h>
//--------------------------------------------------------
typedef int DType;
int cmp(int d1, int d2) {
    if (d1 == d2) return 0;
    return (d1 < d2 ? -1 : 1);
}
//--------------------------------------------------------
struct SACollection {
    DType *data;
    int length;
    int size;
};
typedef struct SACollection *ACollection;

ACollection newACollection(int length) {
    ACollection col = (ACollection) malloc(sizeof(struct SACollection));
    col->data = (DType *)malloc(length * sizeof(DType));
    col->length = length; col->size = 0;
    return col;
}
void freeACollection(ACollection col) {
    free(col->data); free(col);
}
//--------------------------------------------------------
int sizeOfACollection(ACollection col) {
    return col->size;
}
void addACollection(ACollection col, DType x) {
    if (col->size == col->length) {
      // ขยายอาเรย์ตรงนี้ ไม่ได้แสดงรายละเอียด
    }
    col->data[col->size] = x;
    col->size++;
}

int removeACollection(ACollection col, DType x) {
    // ลบ x ออกจาก col (ไม่ได้แสดงรายละเอียด)
}
//============================================================
void removeAny(ACollection col) {
    // ลบข้อมูลตัวใดก็ได้หนึ่งตัว ออกจาก col
}
//============================================================
void containsAny(ACollection col1, ACollection col2) {
    // คืน 1 ถ้ามีสมาชิกสักตัว (หรือมากกว่าก็ได้) ของ col2 ที่เป็นสมาชิกของ col1 ด้วย ถ้าไม่พบเลย คืน 0
}
//============================================================
```

ถ้า **col** เก็บข้อมูล $n$ ตัว
**removeAny(col)** ใช้เวลาเป็น _____

ถ้า **col1** และ **col2** เก็บข้อมูล
$n_1$ และ $n_2$ ตัว ตามลำดับ
**containsAny(col1,col2)** ใช้เวลาเป็น

5. (15 คะแนน) จงออกแบบโครงสร้างข้อมูลที่มีเวลาการทำงานตามความต้องการที่ระบุไว้ด้านล่างนี้ **นิสิตต้องอธิบายรายละเอียดมากพอที่จะนำไปเขียนโปรแกรมได้ โดยไม่จำเป็นต้องเขียนโปรแกรม** (Hint: อนุญาตให้นิสิตระบุว่าจะต้องเรียกใช้คำสั่งเรียงลำดับข้อมูล ซึ่งใช้เวลา O(N log N) ได้ แต่ต้องระบุให้ชัดเจนว่าเรียงจากมากไปน้อย หรือ น้อยไปมาก และ นิสิต**ไม่จำเป็น**ต้องคำนึงถึงกรณีที่ความจุของอาเรย์ไม่พอ และ กรณีที่มีข้อมูลซ้ำในข้อนี้)

5.1 โครงสร้างข้อมูล MaxList มีบริการดังต่อไปนี้ (5 คะแนน)

- MaxList Build_MaxList_from_Array(int a[], int N) สร้าง MaxList จากอะเรย์ของตัวเลข **ทำงานใน O(N log N)**
- void Insert_MaxList(MaxList list, int x) เพิ่มตัวเลข x เข้าไปใน MaxList **ทำงานใน O(N)**
- int DeleteMax_MaxList(MaxList list) ลบตัวเลข**ที่มีค่ามากที่สุด**ใน MaxList แล้วคืนค่า **ทำงานใน O(1)**

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

จงอธิบายหลักการทำงานคร่าว ๆ อธิบายตัวแปรที่จะใช้เก็บข้อมูล พร้อมทั้งวาดรูปประกอบ

อธิบายการทำงานของ `MaxList Build_MaxList_from_Array(int a[], int n)`

อธิบายการทำงานของ `void Insert_MaxList(MaxList list, int x)`

อธิบายการทำงานของ `int DeleteMax_MaxList(MaxList list)`

5.2 โครงสร้างข้อมูล MedianList มีบริการดังต่อไปนี้ (10 คะแนน)

- MedianList Build_MedianList_from_Array(int a[], int N) สร้าง MedianList จากอะเรย์ของตัวเลข **ทำงานใน O(N log N)**
- void Insert_MedianList(MedianList list, int x) เพิ่มตัวเลข x เข้าไปใน MedianList **ทำงานใน O(N)**
- void DeleteMedian_MedianList(MedianList list, int* m1, int* m2) หากใน list มีจำนวนข้อมูลเป็นเลขคี่ บริการนี้จะลบข้อมูลที่ลำดับที่ N/2 ไปเก็บใน *m1 และ *m2(ตัวซ้ายอยู่ลำดับที่ 0) ถ้า N เป็นคู่ ให้ลบตัวที่ N/2 กับ 1+(N/2) ไปเก็บใน *m1 และ *m2 ตามลำดับ **ทำงานใน O(1)**

จงอธิบายหลักการทำงานคร่าว ๆ อธิบายตัวแปรที่จะใช้เก็บข้อมูล พร้อมทั้งวาดรูปประกอบ

จงอธิบายการทำงานของ `MedianList Build_MedianList_from_Array(int a[], int N)`

จงอธิบายการทำงานของ `void Insert_MedianList(MedianList list, int x)`

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

จงอธิบายการทำงานของ `void DeleteMedian_MedianList(MedianList list, int* m1, int* m2)`

6. (10 คะแนน) โครงสร้างข้อมูลประเภทกองซ้อนนั้น ใน struct SStack มีตัวแปรอยู่สามตัวคือ data, size และ length จงเขียนโครงสร้างข้อมูลประเภทกองซ้อนขึ้นใหม่ โดยมีข้อจำกัดคือ กำหนดให้ใน struct SStack นั้นมีเพียงโครงสร้างข้อมูลประเภท BHeap แบบ max heap และ ตัวแปรประเภท int อยู่ 1 ตัวเท่านั้น โดยนิสิตจะต้องเขียนฟังก์ชัน newStack, push และ pop โดยใช้งาน BHeap นี้ นอกจากนี้ นิสิตสามารถใช้งาน BHeap ได้เฉพาะการเรียกฟังก์ชันต่าง ๆ ได้แก่ newBHeap, isEmptyBHeap, sizeOfBHeap, addBHeap, และ removeMaxBHeap โดยห้ามเรียกใช้ตัวแปร data, size, front, length โดยตรงโดยเด็ดขาด การใช้งาน BHeap นั้น **หากนิสิตต้องการเปลี่ยน DType ของ BHeap ให้ไม่เหมือน DType ของ Stack นิสิตจะต้องเขียนโปรแกรมเพื่ออธิบาย DType และฟังก์ชัน cmp ที่จำเป็นด้วย** นิสิตสามารถเขียนฟังก์ชันอื่น ๆ เพิ่มเติมได้ เพื่อความสะดวก โจทย์กำหนดให้ DType ของกองซ้อนเป็น int และมีฟังก์ชัน cmp อยู่เรียบร้อยแล้ว กำหนดให้ DType ของ Stack มีชื่อเป็น DType1 ส่วน DType ของ BHeap เป็น DType2 (หมายเหตุ: ไม่จำเป็นต้องตรวจสอบกรณีที่มีการใส่ข้อมูลเมื่อกองซ้อนนี้เต็ม หรือการเอาข้อมูลออกเมื่อกองซ้อนนี้ไม่มีข้อมูล)

```
// กำหนดให้มีโครงสร้างข้อมูลประเภท BHeap อยู่เรียบร้อยแล้ว ไม่ได้เขียนไว้ ณ ที่นี้ แต่สามารถเรียกใช้ได้ตามปรกติ
// นิสิตจะต้องเขียนโปรแกรมกำหนด DType2 ของ Heap พร้อมทั้งฟังก์ชัน cmp ไว้ตรงนี้ด้วย ถ้าต้องการประกาศฟังก์ชันเพิ่มเติมสามารถเขียนได้ที่ตรงนี้









struct SStack {
  BHeap bh;    // BHeap เป็น max heap ใช้ DType2
  int var1;    // ห้ามประกาศตัวแปรใด ๆ เพิ่มเติม
};
typedef struct SStack *Stack;
Stack newStack(int length) {




}
void push(Stack s, DType1 x) {



}
DType1 pop(Stack s) {









}
```

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

7. (15 คะแนน) เมทริกซ์มากเลขศูนย์ (Sparse Matrix) คือ เมทริกซ์ที่มีจำนวนช่องที่ไม่ใช่ 0 น้อยเมื่อเทียบกับจำนวนช่องทั้งหมดเช่น เมทริกซ์ขนาด NxM (N แถว, M คอลัมน์) จะมีจำนวนช่องทั้งหมด NM ช่อง แต่ เมทริกซ์มากเลขศูนย์ จะมีจำนวนช่องที่ไม่ใช่ 0 น้อยมากๆ (โดยส่วนใหญ่จะเป็น O(N + M) ) วิธีการหนึ่งในการเก็บข้อมูลของเมทริกซ์มากเลขศูนย์ คือ ใช้ Array List จำนวน N AList โดยที่ AList แต่ละอันแทนข้อมูลในแต่และแถวของเมทริกซ์ และใน AList นั้นเก็บข้อมูลเพียงแค่ตำแหน่งที่มีค่าที่ไม่ใช่เลข 0 เท่านั้น ซึ่งจะทำให้ประหยัดเนื้อที่ได้มาก ในข้อนี้นิสิตจะต้องเขียนโปรแกรมภาษา C ของ โครงสร้างข้อมูลเมทริกซ์มากเลขศูนย์ด้วยวิธีนี้ โดยเติมในที่ว่างข้างล่าง

การระบุตำแหน่งของช่องใน เมทริกซ์จะต้องระบุแถว (มีค่าได้ตั้งแต่ 0 ถึง N-1) และ คอลัมน์ (มีค่าได้ตั้งแต่ 0 ถึง M-1)

Hint: นิสิตสามารถเรียกใช้บริการต่างๆของ Array List ที่เรียนในวิชานี้ได้ดังนี้

```
struct SMatrixElement {
      int col;              /*  อยู่คอลัมน์ไหน  */
      float value;      /*  มีค่าเท่าไหร่   */
};     /* ใช้เก็บค่าแต่ละตัวใน Sparse Matrix ที่ไม่ใช่ 0 */
typedef struct SMatrixElement *DType;
/* มี Code ของ Array List อยู่ตรงนี้ แต่ไม่ได้แสดงให้ดู */
struct SSparseMatrix {
    int N, M;     /* จำนวนของแถว (N) และ คอลัมน์ (M) */
    AList* rows;    /* แต่ละแถวของ Sparse Matrix เป็น ArrayList ของ MatrixElement */
};
Typedef struct SSparseMatrix *SparseMatrix;
SparseMatrix newSparseMatrix(int N, int M) {
     SparseMatrix mat; int i;
     mat = (SparseMatrix) malloc(sizeof(struct SSparseMatrix));
     mat->N = N; mat->M = M;
     mat->rows = (AList*)malloc(sizeof(AList)*N);
     for (i = 0; i < N; i++) mat->rows[i] = newAList(10);
     return mat;
}
int cmp(DType a, DType b) {
   /* (1 คะแนน) คืนค่า 0 ถ้าเห็นว่า a เทียบเท่ากับ b มีฉะนั้นคืนค่าที่ไม่ใช่ 0  */




}
float getValue(SparseMatrix mat, int row, int col) {
  /* (3 คะแนน) คืนค่าของ mat ที่ แถว row คอลัมน์ col  */









}
void setValue(SparseMatrix mat, int row, int col, float value) {
   /* (4 คะแนน) กำหนดค่าของ mat ที่ แถว row คอลัมน์ col เป็น value */









}
```

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

```
void multiply(SparseMatrix mat, const float in[], float out[]) {
    /* (7 คะแนน) ทำการคูณ Matrix mat กับ Vector ที่เก็บใน in โดย out[i] จะมีค่าคือ คือ ∑_{j=0}^{n-1} mat[i][j] * in[j]  code จะต้องถูกต้องตามภาษา C และ จะต้องไม่นำช่องที่มีค่าเป็น 0 ใน mat มาคำนวณ เนื่องจากจะช้าเกินไปเมื่อ N และ M มีขนาดใหญ่มาก*/




}
```

8. (14 คะแนน) โครงสร้างข้อมูลประเภทแถวคอย (Queue) นั้นจะใส่ข้อมูลได้ด้านเดียว และข้อมูลจะออกจากอีกด้านหนึ่ง ในข้อนี้เราจะพิจารณาโครงสร้างข้อมูลแบบใหม่ที่เรียกว่า Dual Queue ซึ่งจะคล้ายกับแถวคอยที่มีแถว 2 แถว เรียกว่า “แถวซ้าย” และ “แถวขวา” แต่มีทางออกเพียงทางเดียว (ลองนึกถึงการแถวในในร้านอาหารที่มีสองแถว แต่มีคนทำอาหารเพียงคนเดียว) โดยหลักการทำงานคือ การ enqueue นั้นจะสามารถเลือกได้ว่าจะเข้าไปที่แถวใด ส่วนการ dequeue นั้นมีกฎคือ **เราจะนำข้อมูลออกจากแถวที่ไม่ใช่แถวที่ถูกนำข้อมูลออกในการ dequeue ในครั้งที่แล้ว ยกเว้นแต่ว่าแถวดังกล่าวไม่มีข้อมูล ก็จะนำข้อมูลออกจากจากแถวที่มีข้อมูลแทน** โดยกำหนดให้การ dequeue ครั้งแรกนั้นจะ dequeue ออกจากแถวแรกเสมอ ตัวอย่างเช่น ถ้าแถวแรกมีข้อมูลเป็น <A, B> และแถวที่สองมีข้อมูลเป็น <X, Y, Z, W> ถ้าเราเรียก dequeue 6 ครั้ง จะได้ข้อมูลออกมาเป็น A, X, B, Y, Z, W ตามลำดับ
จงเขียนโปรแกรมสำหรับโครงสร้างข้อมูลชนิดนี้ โดยกำหนดให้ DType ของ Dual Queue นี้เป็นประเภท int (Hint: นิสิตสามารถใช้โครงสร้างข้อมูลอื่น ๆ ที่ได้เรียนมาใช้งานได้ แต่ต้องระบุให้ชัดเจนว่าใช้อะไร และมี DType เป็นอย่างไร)

```
typedef int DType;
struct SDualQueue {



};
typedef struct SDualQueue *DualQueue;
DualQueue newDualQueue(int length) {




}
void enqueue(DualQueue d, int left, DType x) {
  /* (6 คะแนน) ใส่ x เข้าไปในแถวซ้าย ก็ต่อเมื่อ left มีค่าไม่ใช่ 0 ถ้า left มีค่าเป็น 0 จะใส่ในแถวขวา */




}
DType dequeue(DualQueue d) {
   /* (6 คะแนน) นำข้อมูลออกจากแถวที่ไม่ใช่แถวที่ถูกนำข้อมูลออกในการ dequeue ในครั้งที่แล้ว ยกเว้นเมื่อแถวดังกล่าวไม่มีข้อมูล ก็จะนำข้อมูลออกจากจากแถวที่มีข้อมูลแทน */









}
int sizeOfDualQueue(DualQueue q) {
  /*  (2 คะแนน) คืนจำนวนข้อมูลในแถวซ้าย รวมกับแถวขวา */



}
```

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

9. (15 คะแนน) กาลครั้งหนึ่ง นานมาแล้ว วิธีการกำหนดผลการเลือกภาคของนิสิตคณะวิศวกรรมศาสตร์มีหลักการดังต่อไปนี้ มีภาควิชาทั้งหมด 12 ภาควิชา กำหนดให้แต่ละภาคนั้นถูกระบุด้วยตัวเลข 0 ถึง 11 ภาควิชาหมายเลข i จะรับนิสิตได้ไม่เกิน cap[i] คน

มีนิสิตจำนวน n คน นิสิตแต่ละคนถูกระบุด้วยหมายเลข 0 ถึง n-1 นิสิตแต่ละคนจะต้องส่งรายการ “ความต้องการเข้าภาค” ซึ่งระบุถึงภาคที่นิสิตคนนั้นต้องการจะเรียน เรียงลำดับจากต้องการมากสุดไปยังต้องการน้อยสุด รายการนี้ประกอบด้วยหมายเลขของภาควิชา จำนวนไม่น้อยกว่า 1 และไม่มากกว่า 12 หมายเลข และหมายเลขแต่ละตัวจะไม่ซ้ำกันเลย

เราจะทำการกำหนดภาคให้นิสิตทีละคน ตามลำดับของคะแนน GPAX จากมากไปน้อย โดยมีหลักการคือเราจะพยายามทำให้นิสิตได้เรียนภาคที่ต้องการมากที่สุดก่อน แต่ถ้าภาคดังกล่าวมีจำนวนนิสิตครบแล้ว เราจะพิจารณาภาคถัดไปตามลำดับของความต้องการของนิสิต ในกรณีที่ภาคที่นิสิตต้องการทั้งหมดนั้นไม่สามารถรับนิสิตได้เลย (เช่น นิสิตระบุมาแค่ว่าต้องการเข้าภาค 1 และ 2 เท่านั้น แต่ทั้ง 2 ภาควิชาเต็มแล้ว) เราจะให้นิสิตคนดังกล่าวเรียนภาคใดก็ได้ที่มีจำนวนที่นั่งอยู่

ในข้อนี้นิสิตจะต้องเขียนฟังก์ชันโปรแกรมภาษา C เพื่อทำการเลือกภาคดังกล่าว โดยกำหนดให้มีข้อมูลนำเข้าเป็นดังต่อไปนี้ n คือจำนวนนิสิต, AList wishlist[] เป็นอาเรย์ของ AList จำนวน n ตัว ซึ่งมี DType เป็น int โดยที่ wishlist[i] คือรายการเลือกภาคของนิสิตหมายเลข i, int GPAX[] เป็นอาเรย์ขนาด n ช่อง โดยที่ GPAX[i] ระบุคะแนน GPAX ของนิสิตหมายเลย i, int cap[12] เป็นจำนวนนิสิตที่รับได้ของแต่ละภาค (cap[0] + cap[1] + ... cap[11] จะมีค่าอย่างน้อย n) ฟังก์ชันดังกล่าวจะต้องตอบผลลัพธ์ของการเลือกภาคโดยเก็บค่าลงในตัวแปร ACollection dept[12] โดยที่ dept[i] นั้นเป็น array collection ที่มี DType เป็น int ซึ่งระบุหมายเลขของนิสิตที่ได้เข้าภาค i

นิสิตสามารถเรียกใช้โครงสร้างข้อมูลต่าง ๆ ที่ได้เรียนมาในชั้นเรียนได้ แต่นิสิตจะต้องระบุ DType พร้อมทั้งเขียนฟังก์ชัน cmp ของ DType ดังกล่าวตามที่จำเป็น (ยกเว้น DType ประเภท int ให้ถือว่ามีให้อยู่แล้ว) **(***ถ้าพื้นที่ไม่พอให้เขียนต่อด้านหลังของแผนนี้เท่านั้น***)**

```
/*  ถ้าต้องการใช้โครงสร้างข้อมูลใด ให้ระบุไว้ตรงนี้ พร้อมทั้งเขียนฟังก์ชัน  cmp ที่จำเป็นด้วย   หรือถ้ามีฟังก์ชันอื่นใด ๆ ให้เขียนไว้ตรงนี้ได้เลย */









void DeptAssign(int n, AList wishlist[], int GPAX[], int cap[12], ACollection dept[12]) {









}
```