มหาวิทยาลัยสงขลานครินทร์
คณะวิศวกรรมศาสตร์

สอบกลางภาค: ภาคการศึกษาที่ 2 **ปีการศึกษา**: 2545

วันที่สอบ: 28 ธันวาคม 2545 **เวลาสอบ**: 13.30 – 16.30 น.

รหัสวิชา: 240-222 **ห้องสอบ**: R201

ชื่อวิชา: Computer Programming Techniques

คำสั่ง: อ่านรายละเอียดของข้อสอบ และคำแนะนำให้เข้าใจก่อนเริ่มทำข้อสอบ

อนุญาต: เครื่องเขียนต่างๆ เช่น ปากกา หรือดินสอ

ไม่อนุญาต: หนังสือ เอกสารใดๆ และเครื่องคิดเลข

เวลา: 3 ชั่วโมง (180 นาที)

คำแนะนำ

- ข้อสอบมี 6 หน้า (ไม่รวมใบปะหน้า) 4 ข้อ คะแนนรวม 90 คะแนน
- คำตอบทั้งหมดจะต้องเขียนลงใน**สมุดคำตอบ**
- ในการเขียนคำตอบ ให้ขึ้นหน้าใหม่เมื่อเริ่มตอบคำถามข้อใหม่
- คำตอบส่วนใดอ่านไม่ออก จะถือว่าคำตอบนั้นผิด
- การเขียนโปรแกรม ให้ใช้ภาษา C (ให้เป็นไปตาม ANSI C)
- อ่านคำสั่งในแต่ละข้อให้เข้าใจก่อนลงมือทำ

คำถามที่ 1 *(25 คะแนน)*

จงเขียน<u>ฟังก์ชัน</u>เพื่อให้ทำงานได้ตามที่กำหนด โดยมีข้อแม้ว่าจะต้องเขียนเป็นฟังก์ชันที่มีเรียกตัวเองซ้ำ **(Recursion)** <u>ห้าม</u>มีการใช้คำสั่งในการวนรอบ (`while, for, do-while`) รวมทั้งห้ามใช้ฟังก์ชันที่อยู่ใน `math.h` ด้วย

1.1 ฟังก์ชัน `print_bin` ใช้สำหรับพิมพ์ค่าตัวเลขจำนวนเต็มออกมาในรูปของเลขฐานสอง โปรโตไทป์ของฟังก์ชันเป็นดังนี้

```
void print_bin(int number);
```

ตัวอย่างเช่น

`print_bin(5);` ผลลัพธ์ (Output) ที่พิมพ์ออกมาทางหน้าจอคือ 101

`print_bin(12);` ผลลัพธ์ (Output) ที่พิมพ์ออกมาทางหน้าจอคือ 1100

<u>หมายเหตุ:</u> ไม่ต้องมีการตรวจสอบค่าของพารามิเตอร์ `number` สมมุติว่าค่าของ `number` มากกว่า 0 เสมอ

<u>คำแนะนำ:</u> ตัวอย่างการแปลงจากเลขฐานสิบให้เป็นเลขสอง เช่น ถ้าต้องการแปลง 12 ให้เป็นเลขฐานสองทำได้ดังนี้

12 / 2 เท่ากับ 6 เศษ 0

6 / 2 เท่ากับ 3 เศษ 0

3 / 2 เท่ากับ 1 เศษ 1

1 / 2 เท่ากับ 0 เศษ 1

เมื่อหารจนได้ผลหารเป็น 0 แล้ว ให้นำเศษจากการหารมาเขียนเรียงกันโดยเริ่มจากเศษตัวสุดท้าย ดังนั้น 12 จะมีค่าเท่ากับ 1100 ในฐานสอง

1.2 ฟังก์ชัน `find_min` ใช้สำหรับหาค่าที่น้อยที่สุดภายในอาร์เรย์ โปรโตไทป์ของฟังก์ชันเป็นดังนี้

```
int find_min(int a[],int first_elem,int last_elem);
```

ฟังก์ชันนี้จะ return ค่าที่น้อยที่สุดในอาร์เรย์ a เปรียบเทียบตั้งแต่ `a[first_elem]` จนถึง `a[last_elem]` ตัวอย่างเช่น

```
int a[6] = {10,2,9,5,7,3};
int min1,min2;
min1 = find_min(a,0,5);
// ค่าของตัวแปร min1 จะได้เท่ากับ 2 (ค่าที่น้อยที่สุดเมื่อเปรียบเทียบค่าตั้งแต่ a[0] ไปจนถึง a[5])
min2 = find_min(num,2,4);
// ค่าของตัวแปร min1 จะได้เท่ากับ 5 (ค่าที่น้อยที่สุดเมื่อเปรียบเทียบค่าตั้งแต่ a[2] ไปจนถึง a[4])
```

<u>หมายเหตุ:</u> ในการเขียนฟังก์ชันนี้ไม่จำเป็นต้องมีการตรวจสอบค่าของ `first_elem` และ `last_elem` สมมุติว่าค่าของ `first_elem` มีค่าไม่เกิน `last_elem` เสมอ

คำถามที่ 2 *(20 คะแนน)*

2.1 จงเขียน<u>ฟังก์ชัน</u>ชื่อ `combine_str` ซึ่งมีโปรโตไทป์ดังนี้

```
void combine_str(char *target,char *first,char *second);
```

ฟังก์ชันนี้จะทำให้สตริงที่ `target` ชี้อยู่ มีข้อความเหมือนกับการนำสตริงที่ `first` และ `second` ชี้อยู่ มาต่อกัน ตัวอย่างเช่น

```
char str[20];
combine_str(str,"Com","puter");
printf("%s",str);     // ข้อความที่พิมพ์ออกมาคือ Computer
```

ในฟังก์ชัน `combine_str` <u>ห้าม</u>มีการเรียกใช้ฟังก์ชันที่อยู่ใน `string.h`

2.2 ส่วนของโปรแกรมที่ปรากฏอยู่ด้านล่างนี้ เป็นฟังก์ชัน main ที่มีการเรียกใช้ฟังก์ชัน `combine_str` ที่นักศึกษาได้เขียนไปแล้วในข้อ 2.1 จงเติมส่วนที่ขาดหายไป เพื่อให้ได้ผลลัพธ์ (Output) ของโปรแกรมตามที่กำหนด

```
int main()
{
   char *combine,*name;
   char temp[50];
   printf("Enter your name: ");
   scanf("%s",temp);

   ..... // (1).

   ..... // (2)

   strcpy(temp,"Your name is ");

   ..... // (3)

   combine_str(combine,temp,name);
   printf("%s\n",combine);
   return 0;
}
```

ผลลัพธ์ของโปรแกรม (ข้อความที่เป็นตัวหนาคือส่วนที่รับมาจากผู้ใช้)

```
Enter your name: Suntichai
Your name is Suntichai
```

คำถามที่ 3 *(20 คะแนน)*

กำหนดให้ข้อมูลชนิดใหม่ชื่อ Complex เพื่อใช้เก็บข้อมูล real part และ imaginary part ของ complex number (จำนวนเชิงซ้อน) ดังนี้

```
typedef struct {
      float real; //ใช้เก็บข้อมูล real part
      float imag;  //ใช้เก็บข้อมูล imaginary part
} Complex;
```

3.1 *(6 คะแนน)*

กำหนดให้ ฟังก์ชัน `add_complex_1()` มี function prototype ดังนี้

```
void add_complex_1(Complex addend1, Complex addend2, Complex *sum);
```

จงเขียนฟังก์ชัน `add_complex_1()` เพื่อใช้ในการบวก complex number โดยกำหนดให้ฟังก์ชัน `add_complex_1()` มีพารามิเตอร์ 3 ตัว คือ พารามิเตอร์ตัวที่ 1 และ 2 เอาไว้ผ่านค่า complex number ที่ต้องการจะบวก ส่วนพารามิเตอร์ตัวที่ 3 ใช้สำหรับเก็บผลลัพธ์

<u>คำแนะนำ</u>

- ส่วน real part ของผลลัพธ์ จะได้จากนำ ค่า real part ของ complex number ตัวที่ 1 บวกกับค่า real part ของ complex number ตัวที่ 2
- ส่วน imaginary part ของผลลัพธ์ จะได้จากนำ ค่า imaginary part ของ complex number ตัวที่ 1 บวกกับค่า imaginary ของ complex number ตัวที่ 2

3.2 *(6 คะแนน)*

กำหนดให้ ฟังก์ชัน `add_complex_2()` มี function prototype ดังนี้

```
Complex*  add_complex_2(Complex addend1, Complex addend2);
```

จงเขียนฟังก์ชัน `add_complex_2()` เพื่อใช้ในการบวก complex number โดยกำหนดให้ฟังก์ชัน `add_complex_2()` มีพารามิเตอร์ 2 ตัว คือ พารามิเตอร์ตัวที่ 1 และ 2 เอาไว้ผ่านค่า complex number ที่ต้องการจะบวก ฟังก์ชัน `add_complex_2()` จะ return pointer to Complex ซึ่งเป็นผลลัพธ์จากการบวก complex number และนักศึกษาจะต้องสร้างตัวแปร Complex แบบ dynamic (dynamic memory allocation) ไว้สำหรับเก็บค่าผลลัพธ์ และให้ตรวจสอบด้วยว่าได้รับหน่วยความจำที่ขอไปหรือไม่

3.3 *(7 คะแนน)*

จงเขียน main function สำหรับไฟล์ชื่อ adder.c เพื่อ ให้ทำงานดังนี้

- อ่านค่า argument จาก command line และตรวจสอบว่าได้รับ argument จาก command line (command line argument) ถูกต้องหรือไม่ ถ้าผลการตรวจสอบจำนวนแล้วพบว่าไม่ถูกต้อง ให้ print error message และหยุดการทำงานของโปรแกรม รายละเอียดของ argument มีดังนี้

  argument ที่ 1 คือ ค่า real part ของ Complex number ตัวแรก

  argument ที่ 2 คือ ค่า imaginary part ของ Complex number ตัวแรก

  argument ที่ 3 คือ ค่า real part ของ Complex number ตัวที่ 2

  argument ที่ 4 คือ ค่า imaginary part ของ Complex number ตัวที่ 2

- เรียกใช้ ฟังก์ชัน `add_complex_1()` และ `add_complex_2()` และให้แสดงผลลัพธ์ออกทางหน้าจอ

<u>ตัวอย่างการรันโปรแกรมและผลลัพธ์</u>

User พิมพ์ ที่ command prompt

```
~>adder 1.2  3.4  5.6   7.8     //1.2 คือ real part ของ complex number 1
                                 //3.4 คือ imaginary part ของ complex number 1
                                 //5.6 คือ real part ของ complex number 2
                                 //7.8 คือ imaginary part ของ complex number 2
```

ผลลัพธ์ที่แสดงทางหน้าจอ

(1.20,3.40)+(5.60,7.80)=(6.80,11.20) // แสดงผลลัพธ์จากการเรียก add_complex_1( )

(1.20,3.40)+(5.60,7.80)=(6.80,11.20) // แสดงผลลัพธ์จากการเรียก add_complex_2( )

//ให้ใช้ %.2f ในการแสดงผล

คำถามที่ 4 *(25 คะแนน)*

กำหนดให้ข้อมูลชนิดใหม่ชื่อ `student_type` เพื่อใช้เก็บข้อมูลของนักเรียน ดังนี้

```
typedef struct {
      int   student_id;
      char  *first_name;
      char  *last_name;
      float gpa;
} student_type;
```

4.1 *(10 คะแนน)*

กำหนดให้ ฟังก์ชัน `find_student()` มี function prototype ดังนี้

```
int find_student(int search_id,student_type class_list[]);
```

จงเขียนฟังก์ชัน `find_student()` เพื่อใช้ในการค้นหานักเรียน กำหนดให้ฟังก์ชัน `find_student()` มีพารามิเตอร์ 2 ตัวคือ search_id คือ student_id ของนักเรียนที่ต้องการจะค้นหาและ อาร์เรย์ class_list จะเป็นอาร์เรย์ของ student_type หรืออาร์เรย์ที่เก็บข้อมูลของนักเรียนทุกคน กำหนดให้ขนาดของอาร์เรย์ class_list คือ SIZE (สมมุติว่าได้กำหนด SIZE ด้วย `#define SIZE 20`) ไว้เรียบร้อยแล้ว) ให้นักศึกษาใช้ SIZE เมื่อต้องการอ้างถึงขนาดของอาร์เรย์ class_list

ฟังก์ชัน find_student( ) จะ return ค่าเป็น index หรือ subscript ของอาร์เรย์เพื่อบอกว่าเจอนักเรียนในลำดับที่เท่าไร แต่ถ้าหากไม่เจอนักเรียนที่มี student_id ตรงกับที่ต้องการหา ฟังก์ชันนี้จะ return ค่าเป็น –1

4.2 *(10 คะแนน)*

กำหนดให้ ฟังก์ชัน `new_name()` มี function prototype ดังนี้

```
void new_name(char *first,char *last,int found,student_type class_list []);
```

จงเขียนฟังก์ชัน `new_name()` เพื่อเปลี่ยนชื่อและนามสกุลของนักเรียน กำหนดให้ พารามิเตอร์ first และ last เป็นชื่อและนามสกุลใหม่ตามลำดับ พารามิเตอร์ found เป็น student_id ของนักเรียนที่ต้องการเปลี่ยนชื่อและนามสกุล และ อาร์เรย์ class_list จะเป็นอาร์เรย์ของ student_type หรือ อาร์เรย์ที่เก็บข้อมูลของนักเรียนทุกคน

กำหนดให้ขนาดของอาร์เรย์ class_list คือ SIZE (สมมุติว่าได้กำหนด SIZE ด้วย `#define SIZE 20` ไว้เรียบร้อยแล้ว) ให้นักศึกษาใช้ SIZE เมื่อต้องการอ้างถึงขนาดของอาร์เรย์ class_list

4.3 *(5 คะแนน)*

กำหนดให้ ฟังก์ชัน `print_student()` มี function prototype ดังนี้

```
void print_student( int index, student_type class_list[] );
```

จงเขียนฟังก์ชัน `print_student()` เพื่อแสดงข้อมูลของนักเรียนออกทางหน้าจอ พารามิเตอร์ index คือ index หรือ subscript ชองอาร์เรย์ เช่น ข้อมูลของนักศึกษาคนแรกที่ถูกเก็บอาร์เรย์จะมี index หรือ subscript เป็น 0 พารามิเตอร์ อาร์เรย์ class_list จะเป็นอาร์เรย์ของ student_type หรือ อาร์เรย์ที่เก็บข้อมูลของนักเรียนทุกคน

กำหนดให้ขนาดของอาร์เรย์ class_list คือ SIZE (สมมุติว่าได้กำหนด SIZE ด้วย `#define SIZE 20` ไว้เรียบร้อยแล้ว) ให้นักศึกษาใช้ SIZE เมื่อต้องการอ้างถึงขนาดของอาร์เรย์ class_list

ตัวอย่างผลลัพธ์ที่แสดงทางหน้าจอ

```
Student Id: 123456
First Name: Paradorn
Last Name: Srichapan
GPA: 4.00
```