CP 111 Computer Programming, 2/2554, Section B01/B02

Programming Homework 07

กำหนดส่งภายในวันจันทร์ที่ 13 ก.พ. 2555 เวลา 23.59 น.

คำเตือน

นิสิตสามารถจับกลุ่มปรึกษาหารือกันได้ แต่นิสิตแต่ละคนในกลุ่มจะต้องลงมือเขียนโปรแกรมด้วยตนเองและส่งโปรแกรมของตนเองโดยไม่ขึ้นอยู่กับสมาชิกในกลุ่ม ***และอย่านำโปรแกรมของตนเองให้กับผู้อื่นดู***

ถ้าตรวจพบว่านิสิตนำงานของคนอื่นมาส่งหรือนิสิตก็อปปี้ แก้ไข ดัดแปลงโปรแกรมของผู้อื่น จะมีโทษคือปรับตก (ได้เกรด E) ในวิชานี้ทันที

หมายเหตุ

1. ข้อกำหนดการตั้งชื่อไฟล์การบ้านคือ xxx_**HW**yy_**Q**zz.* โดยที่ xxx = รหัสสามตัวท้ายของเลขประจำตัว, yy = หมายเลข homework, zz = หมายเลขข้อในแต่ละ homework (ตัวอย่างเช่น ไฟล์โปรแกรม ข้อ 1 การบ้านครั้งที่ 1 ของนิสิตรหัส 000 คือ 000_HW01_Q01.c) และในแต่ละข้อ

2. การส่งการบ้าน ให้ส่งไฟล์โปรแกรมไปตรวจกับระบบ Café-grader ที่ http://cslab.swu.ac.th/grader

3. การเขียนโปรแกรมที่คำนวณกับตัวเลขที่มีจุดทศนิยม ให้ประกาศตัวแปรที่เกี่ยวข้องเป็นแบบ `double` (Double-precision floating point)

1. Problem name: `Sort3_PassByRef`

(Pointer: Pass-By-Reference)

จงทำโปรแกรมข้างล่างให้สมบูรณ์โดยการเขียนโค้ดในฟังก์ชัน Sort3 ซึ่งจะทำการสลับค่าในตัวแปรที่แอดเดรสของมันถูกส่งมาเป็นอาร์กิวเมนต์ pX1, pX2, pX3 เพื่อทำให้ค่าในตัวแปรทั้ง 3 เรียงจากน้อยไปมาก

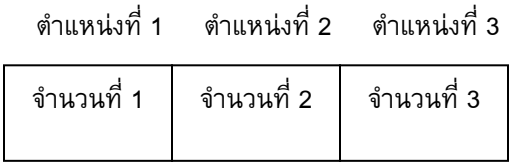
นั้นคือในตัวอย่างโปรแกรมข้างล่าง หลังจากเรียกฟังก์ชัน Sort3 โดยส่งแอดเดรสของตัวแปร a, b และ c มาเป็นอาร์กิวเมนต์ หลังจากฟังก์ชันทำงานเสร็จค่าในตัวแปร a b และ c จะสลับกัน โดยที่ a<=b<=c

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


int main()
{
int a,b,c;

      scanf("%d %d %d",&a,&b,&c);   // <- a, b and c are any values

      Sort3(&a,&b,&c);

      printf("%d %d %d\n",a,b,c);   // <- after Sort3

                                     // a <= b  <= c
      return 0;

}

void Sort3(int *pX1, int *pX2, int *pX3)
{



}
```

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
5 3 -1	-1 3 5
0 -5 2	-5 0 2

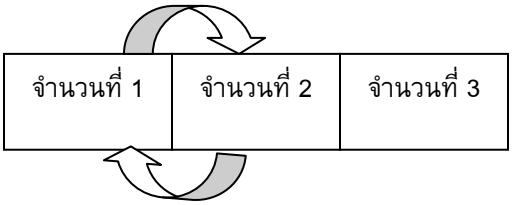
Hint: อัลกอริทึมการเรียงข้อมูลแบบ bubble sort



ตัวอย่าง เช่น 7 5 1

ขั้นตอนที่ 1: เปรียบเทียบค่าของจำนวนที่อยู่ในตำแหน่งที่ 1 กับ ค่าของจำนวนที่อยู่ในตำแหน่งที่ 2

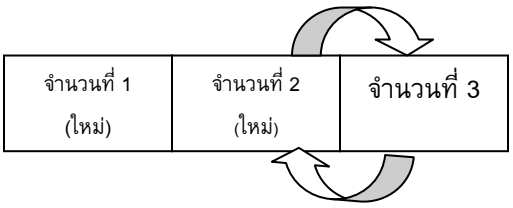
- ถ้าค่าของจำนวนที่อยู่ในตำแหน่งที่ 1 มีค่ามากกว่า ค่าของจำนวนที่อยู่ในตำแหน่งที่ 2 ให้สลับจำนวนที่อยู่ในตำแหน่งที่ 1 และจำนวนที่อยู่ในตำแหน่งที่ 2 (ดูรูปข้างล่าง)



จาก 7 5 1 -> 5 7 1

ขั้นตอนที่ 2: เปรียบเทียบค่าของจำนวนที่อยู่ในตำแหน่งที่ 2 (หลังจากผ่านขั้นตอนที่ 1) กับ ค่าของจำนวนที่อยู่ในตำแหน่งที่ 3

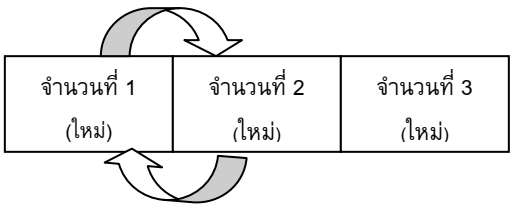
- ถ้าค่าของจำนวนที่อยู่ในตำแหน่งที่ 2 มีค่ามากกว่า ค่าของจำนวนที่อยู่ในตำแหน่งที่ 3 ให้สลับจำนวนที่อยู่ในตำแหน่งที่ 2 และจำนวนที่อยู่ในตำแหน่งที่ 3



จาก 5 7 1 -> 5 1 7

ขั้นตอนที่ 3: เปรียบเทียบค่าของจำนวนที่อยู่ในตำแหน่งที่ 1 (หลังจากผ่านขั้นตอนที่ 2) กับ ค่าของจำนวนที่อยู่ในตำแหน่งที่ 2 (หลังจากผ่านขั้นตอนที่ 2)

- ถ้าค่าของจำนวนที่อยู่ในตำแหน่งที่ 1 มีค่ามากกว่า ค่าของจำนวนที่อยู่ในตำแหน่งที่ 2 ให้สลับจำนวนที่อยู่ในตำแหน่งที่ 1 และจำนวนที่อยู่ในตำแหน่งที่ 2



จาก 5 1 7 -> 1 5 7

เสร็จ

2. Problem name: `Baht_Change`

(Pointer: Pass-By-Reference)

จงทำโปรแกรมข้างล่างให้สมบูรณ์โดยการเขียนโค้ดในฟังก์ชัน Baht_Change ซึ่งมีฟังก์ชันอาร์กิวเมนต์ทั้งหมด 4 ตัว และฟังก์ชันจะไม่คืนค่าอะไร โดยที่ฟังก์ชันอาร์กิวเมนต์ตัวแรก เป็นค่าแบบ int ซึ่งเป็นจำนวนเงินในหน่วยบาท ส่วนฟังก์ชันอาร์กิวเมนต์ตัวที่ 2-10 จะเป็นการส่งค่าแบบ pass by reference ซึ่งจะมีชนิดข้อมูลแบบ pointer to an *int* ซึ่งจะเป็นค่าของ address ของตัวแปรใช้เก็บจำนวนธนบัตรและเหรียญที่มีค่าต่างๆ (1000, 500, 100, 50, 20, 10, 5, 2 และ 1 บาท) ที่แทนจำนวนเงินดังกล่าว

```
#include <stdio.h>
void Baht_Change(      );
int main()
{
int i_money,B1000,B500,B100,B50,B20,C10,C5,C2,C1;
      scanf("%d",&i_money);
      Baht_Change(i_money,&B1000, ...) ;
      printf("%d\n",B1000);
      printf("%d\n",B500);
      printf( );
      ...

      return 0;
}

void Baht_Change(int money, int *B1000, int *B500, int *B100,

                 int *B50,int *B20, int *C10, int *C5, int C2,int *C1)
{
}
```

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
5798	5 1 2 1 2 0 1 1 1

3. Problem name: `Pos_Neg_Avg_Std_Func`

```
void Pos_Neg_Avg_Std_Func(int *x, int N, double *pPosAvg, double * pPosStd,
                                        double *pNegAvg, double * pNegStd)
```

จงเขียนและเรียกใช้ฟังก์ชัน `Pos_Neg_Avg_Std_Func` เพื่อสร้างโปรแกรมที่ทำการรับค่าเลขจำนวนเต็มที่มีค่าไม่เท่ากับศูนย์ (Nonzero integers) ผ่าน keyboard เข้ามาเรื่อยๆ โดยจะหยุดการรับค่าก็ต่อเมื่อผู้ใช้ป้อนค่า 0 โดยสมมติว่าข้อมูลจะมีไม่เกิน 100 จำนวนหลังจากนั้นโปรแกรมจะแสดงค่าเฉลี่ย (average) และค่าเบี่ยงเบนมาตราฐาน (standard deviation) ที่คำนวณจากจำนวนบวกทั้งหมด (positive numbers) และค่าเฉลี่ยและค่าเบี่ยงเบนมาตราฐานที่คำนวณจากจำนวนลบทั้งหมด (negative numbers) แล้วแสดงออกสู่หน้าจอ โดยให้แสดงค่าเฉลี่ยและค่าเบี่ยงเบนมาตราฐานของจำนวนบวกที่บรรทัดแรก แล้วจึงตามด้วยค่าที่คำนวณได้ของจำนวนลบในบรรทัดที่สอง โดยให้แสดงจุดทศนิยม 2 ตำแหน่ง

Note: ค่าเบี่ยงเบนมาตราฐาน (Standard deviation) ของข้อมูล $x_1, x_2, ..,x_N$ คำนวณโดยใช้สูตรข้างล่าง

$$STD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_i - \overline{x}\right)^2}$$

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
-1 10 -5 15 -3 -7 5 0	10.00 4.08 -4.00 2.24
15 -16 17 -18 19 -20 21 22 -23 0	18.80 2.56 -19.25 2.59

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


int main()
{
int N,x;
int data[100];
double pos_avg,pos_std,neg_avg,neg_std;

      N=0;
      do
      {
            scanf("%d",&x);
            if(x!=0)
            {
                ...
            }
      }while(x!=0);

      Pos_Neg_Avg_Std_Func(...);  // <- Function call

      printf("%.2f %.2f\n",pos_avg,pos_std);
      printf("%.2f %.2f\n",neg_avg,neg_std);

      return 0;
}

void Pos_Neg_Avg_Std_Func(int *x, int N,
                          double *pPosAvg, double *pPosStd,
                          double *pNegAvg, double *pNegStd)
{

      // Calculate Mean
      *pPosAvg = ...;
      *pNegAvg = ...;

      // Calculate Standard deviations
      ...
      *pPosStd =...;
      *pNegStd =...;
}
```

4. Problem name: `Reverse_String`

จงเขียนโปรแกรมรับข้อความ (string) จากคีย์บอร์ดจนกว่าผู้ใช้จะกดปุ่ม Enter หลังจากนั้นให้แปลงข้อความนั้นให้สลับจากท้ายไปหน้า แล้วแสดงออกสู่หน้าจอ โดยสมมติว่าจำนวนตัวอักษรที่ป้อนเข้ามาจะไม่เกิน 1000 ตัวอักษร

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
A1B2C3	3C2B1A
Hi, my name is C.	.C si eman ym ,iH
live not on evil	live no ton evil

Hint:

```
#include <stdio.h>
#define MAX_STR_LEN 500
void ReverseStr(char *instr,char *outstr);
int main()
{
char str1[MAX_STR_LEN+1];
char str2[MAX_STR_LEN+1];

      gets(str1);
      ReverseStr(str1,str2);
      printf("%s\n",str2);
      return 0;
}

void ReverseStr(char *instr,char *outstr)
{
int i,k,last;

      i=0;
      while(.......)
            .......
      last=i-1; // ทำให้ last เก็บ index ของ Null character ในอาร์เรย์ instr..[]

      k=0;  // index k จะวิ่งจาก 0, 1, 2...
      for(i=last;i>=0; .......) // index i จะวิ่งจาก last, last-1, last-2, ,1, 0
      {       // copy ตัวอักษรตัวที่ i ใน instr ไปเก็บที่ ตัวอักษรตัวที่ k ใน outstr
            outstr[.....]=instr[......];
            .......
      }
      ....... //  เติม Null character ต่อท้าย outstr
}
```

5. Problem name: `Insert_string`

จงเขียนโปรแกรมเพื่อแทรก (Insert) ข้อความ (String) หนึ่งๆ ลงในข้อความที่ต้องการ ณ ตำแหน่งที่กำหนด โปรแกรมจะรับข้อมูลนำเข้า 3 ค่า คือ (1) ข้อความนำเข้า, (2) ข้อความที่จะนำไปแทรกในข้อความนำเข้า, และ (3) ตำแหน่งในข้อความนำเข้าที่จะแทรก (มีค่าเท่ากับ 0, 1, 2, …) โดยโปรแกรมจะแสดงผลลัพธ์เป็นข้อความที่ผ่านการประมวลผลออกสู่หน้าจอ

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)

`CP111awesome`
`_is_`
`5`
	`CP111_is_awesome`

`EFGH`
`abcd`
`0`
	`abcdEFGH`

`EFGH`
`abcd`
`4`
	`EFGHabcd`

หมายเหตุ

(1) ในชุดทดสอบแแรกเป็นการระบุให้ แทรกสตริง _is_ ลงในสตริง CP111awesome เริ่มที่ตัวอักษรที่ 5 (ซึ่งคือตัวอักษร a) จึงได้ข้อความ CP111_is_awesome

(2) ในชุดทดสอบแแรกเป็นการระบุให้ แทรกสตริง abcd ลงในสตริง EFGH เริ่มที่ตัวอักษรที่ 0 (ซึ่งคือตัวอักษรแรก) จึงได้ข้อความ abcdEFGH

Hint:

```
#include <stdio.h>

void Insert_String(char *instr, char *Ins, int pos,char *outstr);

#define MAX_STR_LEN 501

int main()
{
char str1[MAX_STR_LEN],inserted[MAX_STR_LEN],str2[MAX_STR_LEN];
int pos;

       gets(str1);
       gets(inserted);
       scanf("%d",&pos);
       Insert_String(str1,inserted,pos,str2);
       printf("%s\n",str2);
       return 0;
}

/*    ฟังก์ชัน Insert_String จะทำการสร้าง outstr ที่เกิดจากการแทรกสตริง Ins ลงในสตริง instr เริ่มต้นที่ตำแหน่ง pos    */
void Insert_String(char *instr, char *Ins, int pos,char *outstr)
{
int i,k;

      /* ก็อปปี instr[..] ตัวที่ 0 ถึง pos-1 ลงใน oustr[…]        */
       k=0;
       for(i=0; …)
       {
              outstr[…]=instr[…];
              …
       }
        /* ก็อปปี Ins ต่อท้าย oustr[…]  จากขั้นตอนที่แล้ว   */
       while(…)
       {
              outstr[…]=Ins[…];
              …
            …
       }
      /* ก็อปปี instr[..] ตัวที่ pos ถึง END of string ต่อท้าย oustr[…]  จากขั้นตอนที่แล้ว       */
       i=…
       while(…)
       {
              outstr[…]=instr[…];
              …
              …
       }
       /* ปิดท้ายตัวอักษร NULL character (0) ลงใน outstr */
       …
}
```

6. Problem name: `Twenty_One`

สมมติมีไพ่อยู่ 10 ใบ ไพ่แต่ละใบมีค่าที่เป็นไปได้ในช่วง 1-10 จงเขียนโปรแกรมเพื่อเลือกไพ่ 3 ใบที่ให้ผลรวมมีมากที่สุดแต่ต้องน้อยกว่าหรือเท่ากับ 21 โดยกำหนดให้ข้อมูลนำเข้าคือข้อมูลจำนวนเต็มที่มีค่าในช่วง 1-10 บรรทัดละ 1 ตัว จำนวน 10 บรรทัด และให้แสดงผลลัพธ์เป็นค่าของไพ่ 3 ใบที่ถูกเลือก โดยเรียงจากน้อยไปมาก และผลรวมของไพ่ทั้งสามใบให้อยู่ในบรรทัดเดียวกัน

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
2 2 4 6 8 10 1 1 5 5	5 6 10 21
10 3 5 3 5 4 3 3 1 4	5 5 10 20

Hint:

```
#include <stdio.h>

void Sort3(int *pX1, int *pX2, int *pX3);

int main()
{
int i,j,k,X[10];
int X1,X2,X3,maxsum,sum;

      for(i=0;i<10;i++)
      {
            scanf(…);
      }

      maxsum=...;
      for(i=0;i<10;i++)
      {
            for(...)
            {
                  for(...)
                  {
                        if(... && ... && ...)
                        {
                              sum=...
                              if(... && ...)
                              {
                                    ...
                                    ...
                                    ...
                                    ...
                              }
                        }
                  }
            }
      }

      Sort3(&X1,&X2,&X3);

      printf("%d %d %d %d\n",X1,X2,X3,maxsum);

      return 0;

}
void Sort3(int *pX1, int *pX2, int *pX3)
{
  //Reuse the function from the problem Sort3
}
```

7. Problem name: `Min_Max_in_Range`

กำหนดให้ข้อมูลนำเข้าคือ ชุดของข้อมูลจำนวนเต็ม {x_i; i=1, 2,.., N} จงเขียนโปรแกรมเพื่อคำนวนหาค่าน้อยสุด (Min) และค่ามากสุด (Max) ในบรรดาข้อมูลที่มีค่ามากกว่าหรือเท่ากับ $\bar{X} - STD$ แต่น้อยกว่าหรือเท่ากับ $\bar{X} + STD$ โดยที่ $\bar{X}$ และ STD คือค่าเฉลี่ย (average) และ ค่าเบี่ยงเบนมาตราฐาน (Standard deviation) ของข้อมูล x_1, x_2, ..,x_N คำนวณโดยใช้สูตรข้างล่าง

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} x_i\,, \quad STD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_i - \bar{X}\right)^2}$$

รูปแบบของข้อมูลนำเข้า เป็นดังนี้ บรรทัดที่ 1 เป็นตัวเลขจำนวนเต็มสมมติว่าเท่ากับ N โดย N จะมีค่าไม่เกิน 100 บรรทัดที่ 2 ถึง บรรทัดที่ N+1 จะเป็นข้อมูลแบบจำนวนเต็มบรรทัดละตัว แทน x_i

โปรแกรมจะต้องแสดงผลลัพธ์เป็นค่าเฉลี่ยและค่าเบียงเบนมาตราฐาน (ทศนิยม 2 ตำแหน่ง) และค่าน้อยสุด และค่ามากสุดที่หาเจอ ที่ละบรรทัด

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
6 -1 10 -5 15 -3 -7	1.50 8.12 -5 -1
8 15 -16 17 -18 19 -20 25 22	5.50 18.45 15 22

<u>หมายเหตุ</u> ในตัวอย่างที่ 1 ค่าที่อยู่ในช่วงที่กำหนด คือ -1, -5 และ -3 ในตัวอย่างที่ 2 ค่าที่อยู่ในช่วงที่กำหนด คือ 15, 17, 19 และ 22

Hint:

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

#define MAX_DATA 100

int main()
{
int X[MAX_DATA];
int i,N,sum,min,max;
double avg,sum2,std;

      scanf("%d",&N);
      for(...)
            scanf(...);
      sum=
      for(...)
      {
            sum=
      }
      avg=...

      sum2=0;
      for(...)
      {
            sum2=...
      }
      std=...

      printf("%.2f\n%.2f\n",avg,std);

      min=(int)(avg+2*std);  // set ค่า min เป็นค่าที่มากกว่าค่าที่อาจจะเป็นค่าที่น้อยสุด
      max=(int)(avg-2*std); // set ค่า max เป็นค่าที่น้อยว่าค่าที่อาจจะเป็นค่าที่มากสุด
      for (...)
      {
            if(... )
            {
                  //  update ค่า max
            }
      }
      printf("%d\n%d\n",min,max);

    return 0;
}
```

8. Problem name: `TrimStar`

จงเขียนโปรแกรมรับข้อความ (string) จากคีย์บอร์ดจนกว่าผู้ใช้จะกดปุ่ม Enter หลังจากนั้นให้ลบตัวอักษร '*' ที่อยู่ตอนหน้าและตอนท้ายของข้อความออกไปหลังจากนั้น ให้แสดงข้อความที่ได้ออกสู่หน้าจอ

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
`**A*man*can*not*do*everingthing**`	`A*man*can*not*do*everingthing`
`***We*are*so**smart****`	`We*are*so**smart`
`We*are*so**smart`	`We*are*so**smart`
`***12345`	`12345`
`12345***`	`12345`

หมายเหตุ โจทย์ในข้อนี้จริงๆ แล้วต้องการให้เขียนโปรแกรมทำการลบ Space ที่อยู่หน้าและหลังข้อความหนึ่งๆ ออกไป เช่น จาก " Pradit Mittrapiyanuruk " ให้เป็น "Pradit Mittrapiyanuruk" ซึ่งมีประโยชน์อย่างมากในการประมวลผลข้อความทั่วไป แต่เนื่องจาก Grader ไม่สามารถแยกแยะ space หน้าและหลังได้ จึงต้องแปลงเป็นโจทย์ดังข้างต้น

Hint

```
#include <stdio.h>
#define MAX_STR_LEN 500

void TrimStar(char *instr1,char *instr2);

int main()
{
char str1[MAX_STR_LEN+1];
char str2[MAX_STR_LEN+1];

      gets(str1);
      TrimStar(str1,str2);
      printf("%s\n",str2);
      return 0;
}
void TrimStar(char *instr,char *outstr)
{
}
```

Hint: ดู paper homework