國立屏東大學 資訊工程系 程式設計(二)

Turnin作業1

- Turnin Code: c.hw1
- Due Date: 2/26 23:59:00 **Hard Deadline**

繳交方式說明

本次Turnin作業包含多個程式題，建議同學可以為這次作業先建立一個資料夾hw1，然後在該資料夾內再為每一題建立一個子資料夾，用以進行每一題的作答以及上傳。每一題的子資料夾名稱已寫於題目前方，請務必依照題目的規定建立子資料夾，例如第1題為p1，第2題為p2，餘依此類推。當我們完成某一個題目的作答後，就可以使用turnin指令將該題的答案上傳。以第1題為例，當我們在p1子資料夾裡完成作答後，就可以回到hw1資料夾，使用以下指令將其上傳：

[3:23 user@ws hw1] `turnin␣c.hw1␣p1↵`

當然，你也可以等到所有題目都完成後，再回到hw1資料夾，使用以下指令將所有題目都加以上傳：

[3:23 user@ws hw1] `turnin␣c.hw1␣.↵`

本文使用　及　代表空白字元與Enter換行字元，並且將使用者輸入的部份使用灰階方式顯示。另外，題目的執行結果中，如果出現(、)、:、;、.與,等符號，皆為英文半形！

---

p1 費伯納西數列

費伯納西數列(Fibonacci sequnce)的第n項的值等於其前兩項的和：
\[F_n=F_{n-1}+F_{n-2}\] 且其前兩項的值被定義為：
$$F_0=0$$以及$$F_1=1$$

請設計C語言程式Fibonacci.c以及其標頭檔Fibonacci.h，以遞迴（recursion）方式設計一個名為fibonacci()的函式，並搭配下列的Main.c程式（你可以在/home/stu/public/c2025f/hw1/p1/Main.c找到所需要的檔案）完成第N項費伯納西數的輸出，其中N為大於等於0的正整數：

```
#include <stdio.h>
#include "Fibonacci.h"
```

```
int main()
{
    int N;
    printf("N=? ");
    scanf("%d", &N);
    printf("F_%d=%d.\n", N, fibonacci(N) );
}
```

你必須完成名為Fibonacci.c與Fibonacci.h的C語言程式，其中分別包含fibonacci()函式的Implementation與其Prototype宣告，fibonacci()函式接生一個整數做為參數，並透過遞迴的方式將其前兩項的和傳回做為運算結果。本題的相關程式將使用以下的Makefile進行編譯：

```
all: Main.c Fibonacci.o
	cc Main.c Fibonacci.o

Fibonacci.o: Fibonacci.c Fibonacci.h
	cc -c Fibonacci.c

clean:
	rm -f *.o *~ *.*~ a.out
```

此題的執行結果如下：

```
[3:23 user@ws hw] ./a.out↵
N=?▲0↵
F_0=0.↵
[3:23 user@ws hw] ./a.out↵
N=?▲1↵
F_1=1.↵
[3:23 user@ws hw] ./a.out↵
N=?▲3↵
F_3=2.↵
[3:23 user@ws hw] ./a.out↵
N=?▲10↵
F_10=55.↵
[3:23 user@ws hw]
```

1. 本題應繳交Fibonacci.c與Fibonacci.h兩個檔案，至於Main.c與Makefile則不需繳交。
2. 本題必須使用遞迴方式，透過fibonacci()函式的遞迴呼叫完成計算，若採用其它方式將不予計分。
3. 由於遞迴程式耗用記憶體空間與處理器資源，且過大的項次將會發生整數型態溢位的問題。因此本題輸入之測試資料將不會超過40。

p2 使用輾轉相除法求最大公因數

請參考下面的Main.c程式:

```
#include <stdio.h>
#include "GCD.h"

int main()
{
    int a, b;
    printf("請輸入兩個整數(a,b)=? ");
    scanf("%d,%d", &a, &b);
    printf("%d與%d的最大公因數為%d\n", a, b, GCD(a,b) );
}
```

你必須完成名為GCD.c與GCD.h的C語言程式，其中分別包含GCD()函式的Implementation與其Prototype宣告。GCD()函式接收兩個整數並使用遞迴的方式實作「輾轉相除法」，以求出兩個整數的「最大公因數(Greatest Common Divisor)」。本題的相關程式將使用以下的Makefile進行編譯：

```
all: Main.c GCD.o
	cc Main.c GCD.o

GCD.o: GCD.c GCD.h
	cc -c GCD.c

clean:
	rm -f *.o *~ *.*~ a.out
```

此題的執行結果如下：

```
[3:23 user@ws hw] ./a.out↵
請輸入兩個整數(a,b)=?▲180,▲27↵
180與27的最大公因數為9
[3:23 user@ws hw] ./a.out↵
請輸入兩個整數(a,b)=?▲540,▲840↵
540與840的最大公因數為60
[3:23 user@ws hw] ./a.out↵
請輸入兩個整數(a,b)=?▲115,▲75↵
115與75的最大公因數為5
[3:23 user@ws hw]
```

1. 本題應繳交GCD.c與GCD.h兩個檔案，至於Main.c與Makefile則不需繳交。
2. 本題必須使用遞迴方式，透過GCD()函式的遞迴呼叫完成計算，若採用其它方式將不予計分。

p3 Ackerman函氏計算

請參考下面的Main.c程式:

```
#include <stdio.h>
#include "Ackerman.h"

int main()
{
    int M,N;
    do
    {
        printf("M,N=? ");
        scanf("%d,%d", &M, &N);
    } while( ((M>3)||(N>10))?printf("Wrong Input!\n"):0);
    printf("Ackerman(%d,%d)=%d.\n", M, N, Ackerman(M,N) );
}
```

你必須完成名為Ackerman.c與Ackerman.h的C語言程式，其中分別包含Ackerman()函式的Implementation與其Prototype宣告。Ackerman()函式的定義如下：

$ Ackerman(m,n) = \left\{ \begin{array}{ll} n+1 & \mbox{if } m=0, \\ Ackerman(m-1,1) & \mbox{if } m>0 \land n=0, \\ Ackerman(m-1, Ackerman(m, n-1) & \mbox{if } m>0 \land n>0. \end{array} \right. $

請使用遞迴的方式實作Ackerman()函式。由於Ackerman()函式成長速度極快，為免使用過多系計中運算資源，本題限制M與N的輸入不能大於3與10。相關程式將使用以下的Makefile進行編譯：

```
all: Main.c Ackerman.o
    cc Main.c Ackerman.o

Ackerman.o: Ackerman.c Ackerman.h
    cc -c Ackerman.c

clean:
    rm -f *.o *~ *.*~ a.out
```

此題的執行結果如下：

```
[3:23 user@ws hw] ./a.out↵
M,N=?▲4,2↵
Wrong▲Input!↵
M,N=?▲5,2↵
Wrong▲Input!↵
M,N=?▲3,211↵
Wrong▲Input!↵
M,N=?▲3,2↵
Ackerman(3,2)=29.↵
[3:23 user@ws hw] ./a.out↵
M,N=?▲3,8↵
Ackerman(3,8)=2045.↵
[3:23 user@ws hw] ./a.out↵
M,N=?▲4,2↵
Wrong▲Input!↵
M,N=?▲2,0↵
Ackerman(2,0)=3.↵
[3:23 user@ws hw]
```

1. 本題應繳交Ackerman.c與Ackerman.h兩個檔案，至於Main.c與Makefile則不需繳交。
2. 本題必須使用遞迴方式，透過Ackerman()函式的遞迴呼叫完成計算，若採用其它方式將不予計分。

p4 兩數交換函式設計

請參考下面的Main.c程式:

```
#include <stdio.h>
#include "Swap.h"

int main()
{
   int x, y, *p;
   printf("x=?");
   scanf("%d", &x);
   printf("y=?");
   scanf("%d", &y);
   swap(&x, &y);
   printf("x=%d y=%d\n", x, y);
}
```

你必須完成名為Swap.c與Swap.h的C語言程式，其中分別包含swap()函式的Implementation與其Prototype宣告□swap()函式接收兩個數值的記憶體位址做為參數，並在函式中將這兩個記憶體位址內所包含的數值進行交換。本題的相關程式將使用以下的Makefile進行編譯：

```
all: Main.c Swap.o
        cc Main.c Swap.o

Swap.o: Swap.c Swap.h
        cc -c Swap.c

clean:
        rm -f *.o *~ *.*~ a.out
```

此題的執行結果如下：

```
 [3:23 user@ws hw] ./a.out↵
 x=?1↵
 y=?2↵
 x=2▲y=1↵
 [3:23 user@ws hw] ./a.out↵
 x=?3↵
 y=?5↵
 x=5▲y=3↵
 [3:23 user@ws hw]
```

請注意本題應繳交Swap.c及Swap.h兩個檔案，至於Main.c與Makefile則不需繳交。

p5 傳回最小值的記憶體位址函式設計

請參考下面的Main.c程式:

```
#include <stdio.h>
#include "Min.h"

int main()
{
   int x, y, *p;
   printf("Please input two numbers: ");
   scanf("%d %d", &x, &y);
   p = min(&x, &y);
   printf("The minimum of %d and %d is %d.\n", x, y, *p);
}
```

你必須完成名為Min.c與Min.h的C語言程式，其中分別包含min()函式的Implementation與其Prototype宣告。min()函式接收兩個整數的記憶體位址做為參數，並將其記憶體位址中所包含的數值最小值者的記憶體位址傳回。本題的相關程式將使用以下的Makefile進行編譯：

```
all: Main.c Min.o
	cc Main.c Min.o

Min.o: Min.c Min.h
	cc -c Min.c

clean:
	rm -f *.o *~ *.*~ a.out
```

此題的執行結果如下：

```
[3:23 user@ws hw] ./a.out↵
Please▲input▲two▲numbers:▲1▲2↵
The▲minimum▲of▲1▲and▲2▲is▲1.↵
[3:23 user@ws hw] ./a.out↵
Please▲input▲two▲numbers:▲19▲12↵
The▲minimum▲of▲19▲and▲12▲is▲12.↵
[3:23 user@ws hw]
```

請注意本題應繳交Min.c及Min.h兩個檔案，至於Main.c與Makefile則不需繳交。

p6 將浮點數四捨五入到指定位數的函式設計

請參考下面的Main.c程式，完成所需的rounding()函式設計(rounding()函式的prototype及implementation請分別寫在Round.h及Round.c檔案裡。):

```
#include <stdio.h>
#include "Round.h"

int main()
{
    double x;
    int p;
    printf("Please input a floating number: ");
    scanf(" %lf", &x);
    printf("What decimal place do you want to round? ");
```

```
    scanf(" %d", &p);
    rounding(&x,p);
    printf("%f\n", x);
}
```

呼叫rounding()函式時必須傳入兩個引數：

- 一個浮點數所在的記憶體位址□double *num
- 一個代表位數的整數值□int pos (本題用以測試的$0\leq$pos$\leq$5)

此函式的作用是將所傳入的記憶體位址內的浮點數數值，四捨五入到小數點後第pos位(假設pos□3則113.641590 – > 113.64200)的運算，且其運算結果必須存放於該記憶體位址內。

本題的相關程式將使用以下的Makefile進行編譯：

```
all: Main.c Round.o
        cc Main.c Round.o

Round.o: Round.c Round.h
        cc -c Round.c

clean:
        rm -f *.o *~ *.*~ a.out
```

此題的執行結果如下：

```
[3:23 user@ws hw] ./a.out↵
Please▲input▲a▲floating▲number:▲113.645192↵
What▲decimal▲place▲do▲you▲want▲to▲round?▲0↵
114.000000↵
[3:23 user@ws hw] ./a.out↵
Please▲input▲a▲floating▲number:▲113.645192↵
What▲decimal▲place▲do▲you▲want▲to▲round?▲1↵
113.600000↵
[3:23 user@ws hw] ./a.out↵
Please▲input▲a▲floating▲number:▲113.645192↵
What▲decimal▲place▲do▲you▲want▲to▲round?▲2↵
113.650000↵
[3:23 user@ws hw] ./a.out↵
Please▲input▲a▲floating▲number:▲113.645192↵
What▲decimal▲place▲do▲you▲want▲to▲round?▲3↵
113.645000↵
[3:23 user@ws hw] ./a.out↵
Please▲input▲a▲floating▲number:▲113.645192↵
What▲decimal▲place▲do▲you▲want▲to▲round?▲4↵
113.645200↵
```

```
 [3:23 user@ws hw]
```

請注意本題應繳交Round.c及Round.h兩個檔案，至於Main.c與Makefile則不需繳交。

p7 分解浮點數的函式設計

請參考下面的Main.c程式:

```
#include <stdio.h>
#include "Float.h"

int main()
{
   int i_part;
   double f_part;
   double d;
   printf("Please input a floating number: ");
   scanf("%lf", &d);
   decompose(d, &i_part, &f_part);
   printf("i_part=%d\n", i_part);
   printf("f_part=%f\n", f_part);
}
```

你必須完成名為Float.c與Float.h的C語言程式，其中分別包含decompose()函式的Implementation與其Prototype宣告，decompose()函式接收一個浮點數值以及兩個數值的記憶體位址做為參數，並在函式中將所傳入的浮點數的整數與小數部份分別寫入到這兩個記憶體位址內。本題的相關程式將使用以下的Makefile進行編譯：

```
all: Main.c Float.o
	cc Main.c Float.o

Float.o: Float.c Float.h
	cc -c Float.c

clean:
	rm -f *.o *~ *.*~ a.out
```

此題的執行結果如下：

```
 [3:23 user@ws hw] ./a.out↵
 Please▲input▲a▲floating▲number:▲3.1415926↵
 i_part=3↵
```

```
f_part=0.141593↵
[3:23 user@ws hw] ./a.out↵
Please▲input▲a▲floating▲number:▲3↵
i_part=3↵
f_part=0.000000↵
[3:23 user@ws hw] ./a.out↵
Please▲input▲a▲floating▲number:▲0.23↵
i_part=0↵
f_part=0.230000↵
[3:23 user@ws hw]
```

請注意本題應繳交Float.c及Float.h兩個檔案，至於Main.c與Makefile則不需繳交。