1051NTOUCSE 程式設計 1C 期中考參考答案

姓名：_____ 系級：_____ 學號：_____ 105/11/09 (三)

考試時間：14:30 – 16:30

考試規則：1. **請闔上課本**，除了印給你的之外**不可**參考任何文件包括小考、作業、實習、或是其他參考資料
2. 你可以在題目卷上直接回答，可以使用鉛筆，但是**請在答案卷上註明題號**
3. 你覺得有需要的話，可以使用沒有教過的語法，但是**僅限於 C 語言**
4. **不可**使用電腦、平板、智慧手機、及工程型計算機
5. 請不要左顧右盼！請勿討論！請勿交換任何資料！對於題目有任何疑問請舉手發問
6. 如果你提早交卷，請迅速安靜離開教室，並請勿在走廊喧嘩
7. 違反上述考試規則視為不誠實的行為，由學校依學務規章處理
8. 請在題目卷及答案卷上**寫下姓名及學號**，交卷時請繳交**題目卷**及**答案卷**
9. 本份考卷共有 **110** 分的題目，但是你得到的成績**超過 100 的部份不算在學期成績裡**

1. [8] 請問 long long 資料型態的變數所存放資料的格式為何？能夠存放的整數範圍為何？程式執行時 sizeof(long long) 的數值為多少？使用 scanf 由鍵盤讀入 long long 型態變數時需要使用什麼格式轉換命令？

Sol.

格式是二的補數，能夠存放的整數範圍在 -2^{63}~2^{63}-1, sizeof(long long)的數值是 8, %lld 或是在 MS windows 下可以用 %I64d

2. [8] 請使用 switch 的語法將整數變數 score 中 0~100 的成績轉換為等第列印出來？ (80~100 為 A, 70~79 為 B, 60~69 為 C, 0~59 為 F，請注意語法，使用五個 case，請不要使用 if/else 敘述)

Sol.

```
switch (score/10) {
case 10: case 9: case 8:
      printf("A\n"); break;
case 7:
      printf("B\n"); break;
case 6:
      printf("C\n"); break;
default:
      printf("F\n");
}
```

3. [8] 如果在鍵盤輸入右方的數字，請問下列程式片段執行完畢後 x, y, z, w 變數內的資料為何？

```
123↵
45↵
678↵
```

```
int x; char y, z; int w;
scanf("%d", &x); scanf("%c%c", &y, &z); scanf("%d", &w);
```

Sol.

x 為 123 (十進位), y 為'\n' (十進位 10), z 為'4' (十進位 52), w 為 5 (十進位)

4. [8] 請撰寫一個 inputData 函數，把上題的三個 scanf 包裝到函數裡面，由鍵盤讀到的資料需要傳回 inputData 的呼叫端 (請注意參數的宣告)，並且寫出呼叫 inputData() 函數的敘述

Sol.

```
void inputData(int *x, int *w, char *y, char *z) {
      scanf("%d", x); scanf("%c%c", y, z); scanf("%d", w);
}
…
int x,w; char y, z;
inputData(&x, &w, &y, &z);
```

5. [8] 請寫一個 while 迴圈，運用泰勒展開式 $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \cdots$ 計算 e^x 直到 $\frac{x^n}{n!} < 10^{-5}$
(請注意運算過程中避免產生溢位，只要單項大於 10^{-5} 的都加起來, -100<x<100, 最後結果希望有十五位有效位數)

Sol.

```
double sum = 1, term, x;
int n = 2;
// scanf("%lf", &x);
while (term>=1e-5) {
        sum += term;
        term = term * x / n;
        n++;
}
// printf("%lf", sum);
```

6. [8] 請列舉四個運用『函數』來組織你的程式的原因

Sol.

1. 程式裡不要出現重複的程式碼，不需要拷貝與貼上，也避免維護的不一致問題
2. 函數可以把邏輯目標相同的程式敘述合在一起，避免不必要的細節干擾程式其它部份的運作，函數可以有一個代表它的功能的名字
3. 函數可以有獨立的變數命名空間，其他函數沒有辦法直接用到函數裡面的變數，函數裡面也不能夠使用其他函數裡的變數，可以降低程式碼的複雜度
4. 函數裡面的邏輯和其他函數的邏輯是無關的

7. 請撰寫一個程式，如右方輸入一個 0～2^{64}-1 範圍內的整數 n，再輸入指定的進位制 m，以 m 進位列印 n，進位制超過 10 時請使用 A 代表 10, B 代表 11, …., Z 代表 35, 請不要使用 math.h 中的函數

```
54321↵
2↵
1101010000110001
```

   a. [5] 請撰寫一個函數尋找 m^k 使得 $m^{k+1} > n \geq m^k$

Sol.

```
unsigned long long int findMaxWeight(unsigned int base, unsigned long long n) {
        unsigned long long int y=1;
        while ((n/=base)>0) y *= base;
        return y;
}
```

```
54321↵
16↵
D431
```

   b. [5] 請撰寫一個函數，傳入一個 0～35 範圍內的整數，列印其代表符號 (0~9, A~Z)

Sol.

```
void output(int digit) {
        if (digit<10)
                printf("%c", digit+'0');
        else // if (digit<36)
                printf("%c", digit-10+'A');
}
```

```
54321↵
36↵
15WX
```

   c. [10] 請撰寫一層迴圈運用上面的函數、除法及求餘數的運算列印每一位數

Sol.

```
unsigned int base;
unsigned long long int number, weight;
scanf("%llu%u", &number, &base);
for (weight= findMaxWeight(base, number); weight>0; weight/=base)
```

```
        output(number/weight%base);
    printf("\n");
```

下面兩題中關於運算量的額外說明:

```
for (i=0, sum=0; i<n; i++)
    sum += data[i];
```

上面這個迴圈所執行的總運算次數正比於 n; 如果你寫一個兩層的迴圈, 兩個迴圈執行的次數都正比於 n, 則程式所做的總運算次數正比於 n^2

8. 質因數只有 2 和 3 的正整數由小到大排列如下 1, 2, 3, 4, 6, 8, 9, 12, 16, 18, … (把 $1=2^0 3^0$ 也算進來)

   a. [5] 請撰寫一個函數 int isRequired(long long int x) 測試一個正整數的質因數是否只有 2 和 3

Sol.

```
int isRequired(long long int number) {
    if (number==1) return 1;
    while (1)
        if (number%2==0)
            number /= 2;
        else if (number%3==0)
            number /= 3;
        else
            break;
    return number==1;
}
```

   b. [5] 請撰寫一小段程式, 讀入一個整數 n, 直接運用迴圈測試每一個正整數來找到上面這個數列中第 n 個元素

Sol.

```
long long int n, i=0;
scanf("%lld", &n);
while (n)
    if (isRequired(++i)) n--;
printf("%lld\n", i);
```

   c. [10] 題 b 中的輸入如果是 10000 就會發現程式需要執行非常久, 因有太多的正整數不在這個數列中, 如果我們能夠只看到這個數列裡的數字, 要找到第 10000 個就會快得很多, 請撰寫一個執行時間與 n 成正比的程式 (請定義一個陣列 long long int a[10001]; 來存放找到的數字, 也就是 a[1]為 1, a[2]為 2, a[3]為 3, a[4]為 4, a[5]為 6, …因為我們曉得每一個數字乘上 2 以及乘上 3 都一定會在這個數列裡, 記錄下來的話, 我們可以根據這些數字來產生下一個數字, 可惜 2*a[1], 3*a[1], 2*a[2], 3*a[2], 2*a[3], 3*a[3], …不一定由小排到大, 如果是的話很容易地就寫完了, 不過我們曉得對於任意 i<j 而言, 2*a[i]<2*a[j]同時 3*a[i]<3*a[j], 所以我們可以運用兩個變數 index2, index3 來紀錄目前列出的元素是 2*a[index2] 或是 3*a[index3], 依序增加 index2 或是 index3 即可找到數列中第 n 個數字, 請特別注意不要重複計算相同的數字)

Sol.

```
long long int n, i=1, index2=1, index3=1, x2, x3;
long long int a[10001];
scanf("%lld", &n);
a[1] = 1;
x2 = a[index2]*2, x3 = a[index3]*3;
```

```
while (++i<=n) {
    if (x2<x3) {
        a[i] = x2;
        x2 = a[++index2]*2;
        if (x2==x3) x2 = a[++index2]*2;
    }
    else { // if (x3<x2)
        a[i] = x3;
        x3 = a[++index3]*3;
        if (x3==x2) x3 = a[++index3]*3;
    }
}
printf("%lld\n", a[n]);
```

9. 有 n 個非負整數 $\{a_i : 0 \leqq i \leqq n-1\}$, $1000 \leqq n \leqq 100000$, $0 \leqq a_i \leqq 32767$, 定義出 n 條與 x-軸垂直的線段, 線段的端點為 (i,0) 與 (i, a_i), 任意兩條線段(i, a_i)與(j,a_j)加上 x-軸可以看成是一個盛水的容器, 可以盛水的容積為$|j-i| * \min(a_i, a_j)$, 假設 n 個整數已經在一個整數陣列 a[] 裡面, 在下列要求下撰寫程式片段找出能夠圍出最大容積的兩個線段, 以及最大容積:

   a. [8] 請使用兩層的迴圈列舉所有可能的 i 與 j, 找到並且印出能夠圍出最大容積的 i, j, 以及最大容積

Sol.

```
int max, maxi, maxj, i, j, tmp;
for (max=i=0; i<n; i++)
    for (j=i; j<n; j++)
        if ((tmp=a[a[i]<=a[j]?i:j]*(j-i))>max)
            max=tmp, maxi=i, maxj=j;
printf("a[%d]:a[%d]=%d:%d (最大容積為%d)\n", maxi, maxj, a[maxi], a[maxj], max);
```

   b. [6] 很不幸地, 題 a 這個兩層迴圈的程式在 n=100000 時需要 30 秒以上的執行時間, 如果 n 更大的話, 執行時間會正比於 n^2, 右圖是一個尋找時間正比於 n 的程式, 請解釋這個演算法為什麼只使用了正比於 n 的尋找時間

```
01  i=0, j=n-1;
02  while (i<=j)
03      if (a[i]<=a[j]) {
04          if ((tmp=(long long)a[i]*(j-i))>max)
05              max=tmp, maxi=i, maxj=j;
06          i++;
07      }
08      else {
09          if ((tmp=(long long)a[j]*(j-i))>max)
10              max=tmp, maxi=i, maxj=j;
11          j--;
12      }
13  printf("a[%d]:a[%d]=%d:%d (amount of water is %lld)\n",
14          maxi, maxj, a[maxi], a[maxj], max);
```

Sol.

右邊這個程式裡面兩個變數 i 一開始是 0, j 一開始是 n-1, 迴圈執行的過程中不是 i++ 就是 j--, 所以基本上迴圈就只執行了 n 次, 所以只使用了正比於 n 的尋找時間

   c. [8] 接題 b, 請簡單證明這個演算法一定可以找到最大容積的兩個線段 (考慮每次 i++ 或是 j—時會跳過哪些可能的線段組合)

Sol.

本來在題 a 中我們可以看到解的空間有 n(n-1)/2 個元素, 題 a 中希望你一個一個去測試, 所以會得到一個運算時間正比於 n^2 的演算法, 題 b 的程式中並沒有把每一個解都測試過, 如果能夠得到正確的答案, 需要滿足一些特別的性質

基本上題 b 的演算法就是『當 a[i]<=a[j] 時把 i 值加 1, 反之把 j 值減 1』,

令容積 V(a[i],a[j]) = (j-i) * min(a[i], a[j]), j>=i, 分別考慮下面兩種狀況

1. 當 a[i]<=a[j] 時, 演算法把 i 值加 1 時, 相當於跳過了 (a[i],a[j-1]), (a[i],a[j-2]), …,

(a[i],a[i+1]) 的這些組合，後續完全不會再檢查這些組合的容積了，實際上因為 a[i]<=a[j]，不論 a[j-1]大於、等於或是小於 a[i]，都可以推論出 V(a[i],a[j-1]) < V(a[i],a[j])，同理可以推論出 V(a[i],a[j-2]) < V(a[i],a[j]), …，以及 V(a[i],a[i+1]) < V(a[i],a[j])，所以跳過這些組合是絕對安全的，不會影響到結果的正確性；至於 (a[i],a[j+1]), (a[i],a[j+2]), …, (a[i],a[n-1])，這些組合不是在先前的步驟中考慮過，就是在先前的步驟中被跳過了

2. 當 a[i]>a[j] 時，演算法把 j 值減 1 時情況也完全一樣