1141 NTOUCSE 程式設計 1C 期中考

姓名：_____ 系級：_____ 學號：_____ **114/10/28** (二)

考試時間：**13:20 – 16:00**

考試規則：1. **請闔上課本**，**不可**參考任何文件包括小考、作業、實習、或是其它參考資料
2. 你可以在題目卷上直接回答，可以使用鉛筆，但是**請在答案卷上寫下題號，並且註明答案在題目卷上**
3. 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. 程式撰寫時請寫完整的程式碼，寫… 的分數很低 (程式裡有重複很多遍的敘述本來就是扣分的)
5. **不可**使用電腦、平板、電子紙、智慧手機、手錶、及工程型計算機
6. 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
7. 如果你提早交卷，請迅速安靜地離開教室，請勿在走廊喧嘩
8. 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
9. 請在**題目卷**及**答案卷**上都**寫下姓名及學號**，交卷時請繳交**題目卷**及**答案卷**

1. (a) [10] 請完成右圖程式讀入如下圖多列的資料直到資料串流結束，其中每一列的第一個字元是 O、H、或是 D 分別代表接下來是 8 進位 13～18 位數的整數、或是 16 進位 10~14 位數的整數、或是 10 進位 13~17 位數的整數，小括號裡面是與前面同樣進位制與範圍的數字、而且不會有任何空格，請在 20 格的空間裡靠左對齊以 10 進位列印出該列兩個數字的差，下右圖中 □ 代表空格，請使用適當的資料型態定義兩個變數 x, y，請運用 scanf 讀入資料，以 printf 輸出兩數的差? (如果你的答案裡有不能省略的空格，請以 □ 標示清楚)

```
O12345676544012□□□(725571123)↲
X4AF31024FC□(2451)↲
D44890231002(9201235)↲
D12139876□□(22456678)↲
```

```
01 #include <_____>
02
03 int main()
04 {
05      _____ x, y;
06      char c;
07      while (1==scanf("_____", _____))
08      {
09          if (_____)
10              scanf("_____", &x, &y);
11          else if (_____)
12              scanf("_____", &x, &y);
13          else
14              scanf("_____", &x, &y);
15          printf("_____\n", x-y);
16      }
17      return 0;
18 }
```

(b) [6] 上面這個程式如果運用字元陣列來存放格式轉換命令的話，可以修改第 05~14 列成為:

```
06      char c, fmt[]="----------"; // 和題 (a) 中第 10、12 或是 14 列的格式轉換字串相同
07      while (1==scanf("----------")) // 和題 (a) 中第 07 列相同
08      {
09          fmt[_____] = fmt[_____] = _____;
10          scanf(_____, &x, &y);
```

```
01 int x=63;
02 double y=5.678;
03 printf("_____\n", x, y);
```

(c) [4] 請完成右圖程式用 printf() 得到左圖的輸出，其中前面 6 位數是 8 進位整數，不足 6 位數時需要補 0 不能有空格，接下來是逗點，以及一個 10 進位浮點數 (整數部份固定 6 位數、小數部份 2 位數、小數第 3 位以後四捨五入)，左圖中 □ 代表空格

```
000077,□□□□5.68
```

(d) [10] 有的時候為了找出邏輯的錯誤，會在迴圈裡列印變數的數值，但是迴圈如果重複的次數很多，螢幕一直捲動而來不及分析印出的資料，如果希望在列印之後由使用者以鍵盤按鍵控制是否繼續，例如右圖程式在執行的時候發現一直不會停下來，所以在 printf 之後加上 getchar() 敘述，但是使用者按下按鍵 'y' 時，如右圖除了看到顯示 y 之外，程式並不會列印下

```
01 double x;
02 for (x=0; x!=3000; x+=0.3)
03 {
04      printf("%f □", x);
05 }
```

```
0.000000 y
```

一個資料，如果接下來使用者按 <enter> 按鍵，如右圖程式會一次印出兩個資料，為什麼？這時如果不修改程式該怎樣按一個按鍵顯示一筆資料？不過這個時候發現程式還是一直困在迴圈裡面，沒有辦法正常結束，如果希望在按下 'a' 按鍵時可以結束程式，該怎樣修改程式？

```
0. 000000 y
0. 300000  0. 600000
```

2. 下列幾個版本的程式目標完全一樣，由鍵盤讀入一個不含空格的英數字字串，例如 xndrf，然後列印出如左圖由不同字元開始旋轉過的字串。請根據要求完成下列各個子題的程式

```
xndrf↵
ndrfx↵
drfxn↵
rfxnd↵
fxndr↵
```

```
01 #include <stdio.h>
02 int main()
03 {
04        char_____;
05        int i, j, len;
06        while (___==scanf("%___%n",  ___, &len))
07              for (i=0; i<len; i++)
08              {
09                    for (j=0; j<len; j++)
10                          printf("%c", buf[____%___]);
11                    printf("\n");
12              }
13        return 0;
14 }
```

(a) [10] 右上圖中程式在第 04 列定義一個能夠容納最多 1000 個字元的 char 陣列 buf，第 06 列運用 scanf 函式讀取輸入的字串到陣列 buf 以及字串的長度到整數變數 len，第 10 列直接根據迴圈控制變數 i 及 j 以及字串長度 len 計算出需要列印的字元位置，請注意 scanf()讀取字串的時候會在字串結束以後放一個 '\0' 字元，此字元的 ASCII 編碼值就是 0。

```
07        for (i=0; i<len; i++)
08        {
09              for (j=0; j<_____; j++)
10                    printf("%c", buf[_____]);
11              for (j=0; j<_____; j++)
12                    printf("%c", buf[_____]);
13              printf("\n");
14        }
```

(b) [4] 右圖程式與題(a)中程式主要差別在於第 09~12 列對於每一個旋轉起始 i 值，使用兩個獨立的迴圈(第 09,10 列以及第 11,12 列)來列印字串的後半段與前半段，請完成這個版本

```
06        while (1==scanf("%s", buf))
07              for (i=0;_____; i++)
08              {
09                    ...
```

(c) [2] 題(a)與題(b)中透過 scanf()在讀取輸入字串時也同時取得輸入字串的長度 len，這個輸入字串的長度當然也可以透過額外迴圈或是 string.h 裡的 strlen()函式算出，但是其實不見得需要使用這個長度資訊 len 來控制程式中的迴圈，請完成上圖這個程式片段來取代題(a)程式中的第 6~8 列，不使用字串的長度來控制迴圈執行的次數

```
07        for (i=0; i<len; i++)
08        {
09              printf("_____","_____");
10              t = buf[i];
11              buf[i] = _____;
12              printf("_____\n","_____");
13              _____;
14        }
```

(d) [6] 題(a)與題(b)中都是透過 printf("%c", …)一個字元一個字元輸出，但是 printf()是可以一次輸出整個字串的，請完成右圖的程式片段，用兩個 printf()敘述來取代題(a)中第 09,10 列的輸出迴圈或是題(b)中第 09~12 列的兩個輸出迴圈，注意圖中第 11 列必須修改 buf 字元陣列中第 i 個元素，才能夠運用 printf()列印字串的功能，但是因為第 09~13 列是在第 07 列迴圈中，如果 buf 字元陣列被破壞掉了，接下來 i++以後的迴圈就沒辦法正常運作了，所以第 10 列需要把 buf[i]的內容先備份到字元變數 t 裡面，第 13 列再還原回去。

```
16              for (i=0; i<len; i++)
17              {
18                    rotate_print1(buf, i, len);
19                    printf("\n");
20              }
```

(e) [4] 前面(a)~(d)都使用迴圈來完成題目的要求，我們知道遞迴函式是第二種表達重複動作的程式語法，遞迴的設計很多樣化，右側程式片段是一個遞迴的設

計，第 16~20 列是 main() 函式裡呼叫遞迴函式的迴圈，取代題(a)中第 07~12 列迴圈的功能，第 02 列遞迴函式的參數 i 是指定本次呼叫由 buf 陣列的第 i 個字元開始列印，參數 c 代表要印出幾個字元，請完成右圖中的程式片段 (提示：遞迴的概念在於印出第 i 個字元後只需要由下一個字元開始印出剩下的字串就完成了)

```
02 void rotate_print1(char buf[], int i, int c)
03 {
04      if (c>0)
05      {
06          printf("%c", buf[i]?buf[i]:buf[i=0]);
07          rotate_print1(buf, _____, _____);
08      }
09 }
```

(f) [4]右側程式片段第 21~25 列是 main()函式裡呼叫遞迴函式的迴圈，取代題(a)中第 07~12 列迴圈的功能，第 02 列遞迴函式的參數 i 是指定本次列印由哪一個字元開始，每次遞迴呼叫只印一個字元，參數 j 代表是第幾次遞迴的呼叫，請完成右圖中的程式片段 (提示：j<i 時需要定義出倒數第 j 個字元在 buf[] 陣列中的位置)

```
21          for (i=0; buf[i]!=0; i++)
22          {
23              rotate_print2(buf, i, 0);
24              printf("\n");
25          }
```

```
02 void rotate_print2(char buf[], int i, int j)
03 {
04      if (j<i)
05      {
06          rotate_print2(buf, i, j+1);
07          printf("%c", buf[_____]);
08      }
09      else if (buf[j]!=0)
10      {
11          printf("%c", buf[_____]);
12          rotate_print2(buf, i, j+1);
13      }
14 }
```

3. 如果 x 是一個 double 的變數，n 是一個非負整數，用 n-1 次的乘法 x * x * … * x 計算 x^n 是很沒效率的，有效率的方法需要做「平方再乘」，兩種計算方法的範例如下：$x^{13} = x^{2^3+2^2+0\cdot 2^1+1} = x^{(((1)\cdot 2+1)\cdot 2+0)\cdot 2+1} = \left(\left((x)^2 \cdot x\right)^2 \cdot x^0\right)^2 \cdot x$ 或是 $x^{13} = x^{1+0\cdot 2^1+2^2+2^3} = (x)^1 \cdot (x^2)^0 \cdot (x^4)^1 \cdot (x^8)^1$

(a) [10] 請完成右下圖的函式，實現第一種計算方法，由於方法中由 n 的最高位元開始計算，所以在下面 04, 05 兩列以 while 迴圈先計算出 $w=2^{m-1}$，其中 m 是 n 的位元數，第 06~10 列以 w 來控制迴圈執行的次數，第 8 列先將目前乘積平方，第 9 列再根據該位數為 0 或 1 決定是否將 n 扣除 w 並且將目前乘積乘 x

(b) [6] 請完成下圖程式以遞迴方式實現題(a)的方法

```
01 double power1(double x, int n)
02 {
03      if (n==0)
04          return 1.;
05      else
06      {
07          double x2 = power1(_____, _____);
08          return (_____?_____:_____) * _____;
09      }
10 }
```

```
01 double power1(double x, int n)
02 {
03      double prod;
04      int n1=n, w=_________;
05      while ((n1/=2)>0) _________;
06      for (prod=1; w>0; w/=2)
07      {
08          _____________;
09          if (_________)
10          {
11              n-=w;
12              _______;
13          }
14      }
15      return prod;
16 }
```

(c) [8] 請完成下面的函式實現第二種計算方法，其中第 04, 05 列迴圈重複次數為 n 的二進位位元數，以 n 的數值來控制，每次除以 2 直到 0 為止，每次迴圈執行時的 x 數值是前一次執行時的平方

```
01 double power2(double x, int n)
02 {
03        double prod;
04        for (prod=1; n_____; n_____)
05        {
06              prod *= _________;
07              x _________;
08        }
09        return prod;
10 }
```

(d) [6] 請完成下圖程式以遞迴方式實現題(c)的方法

```
01 double power2(double x, int n)
02 {
03        if (n==0)
04              return 1.;
05        else
06              return _________* power2(_________, _________);
07 }
```

4. (a) [4] 下面程式裡有一個 double 陣列 values，裡面存放了 n 筆浮點數資料，程式第 08 列打算使用 stdlib 函式庫裡面的 qsort() 工具函式把陣列裡面的資料由小到大排好，請完成這個程式：

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 int compare(const void *pa, const void *pb) { return *(double *)pa-*(double *)pb; }
04 int main()
05 {
06        int i, n=10;
07        double values[] = {40.3, 10.2, 10.8, 10.4, 11.1, 10.2, 100.9, 90.1, 20.2, 25.4};
08        qsort(_________,_________,_________,_________);
09        for (i=0; i<n; i++)
10              printf("%6.1f ", values[i]);
11        printf("\n");
12        return 0;
13 }
```

(b) [2] qsort 這個工具函式的原型 (prototype) 如下，請問第四個參數的型態我們稱它為什麼?

```
void qsort(void* base, size_t num, size_t size, int (*compare)(const void*,const void*));
```

(c) [2] 上面程式執行的結果如下：

```
  10.2    10.8    10.4    11.1    10.2    20.2    25.4    40.3    90.1   100.9
```

這個結果的順序好像不太對，請問是什麼原因?

(d) [2] 請完成下列的程式修改：

```
01 int compare(const void *pa, const void *pb)
02 {
03        double *pa1 = _________pa, *pb1 = _________pb;
04        if (*pa1 < *pb1)
05              return _________;
06        else if (*pa1 == *pb1)
07              return _________;
08        else
09              return _________;
10 }
```