1041 國立台灣海洋大學資訊工程系 1C　程式設計 期中考參考答案

姓名：_____

系級：_____

學號：_____

104/11/04 (三)

考試時間：**13:20 – 16:00**

請儘量回答，總分有 **131**，看清楚每一題所佔的分數再回答

考試規則：1. 不可以翻閱參考書、作業及程式
2. 不可以使用任何形式的電腦 (包含手機、計算機、相機以及其它可運算或是連線的電子器材)
3. 請勿左顧右盼、請勿交談、請勿交換任何資料、試卷題目有任何疑問請舉手發問 (看不懂題目不見得是你的問題,有可能是中英文名詞的問題)、最重要的是隔壁的答案不見得比你的好，白卷通常比錯得和隔壁一模一樣要好
4. 提早繳卷同學請直接離開教室，請勿逗留喧嘩
5. 違反上述任何一點之同學一律依照學校規定處理
6. 繳卷時請繳交***簽名過之試題卷及答案卷***

1. [6] 請問如果你的程式檔案名稱使用 xxx.cpp 和 xxx.c 有什麼差別?

   Sol:

   xxx.cpp 是告訴編譯器這個檔案裡面的程式請用 C++ 語法編譯, xxx.c 則是告訴編譯器這個檔案裡面的程式請用 C 語法編譯，有一些語法 C++ 中有比較簡化一點的語法，例如變數可以在任何地方定義

2. [6] 請問 char, long, double 三種資料型態的變數所存放資料的格式各為何? 各使用幾個位元組的記憶體 (byte) ?

   Sol:

   char 和 long 定義出來的變數所存放資料都是二的補數的整數, char 用 1 個位元組的記憶體, long 用 4 個位元組的記憶體; double 存放的資料是 IEEE754 的倍精準浮點數格式，使用 8 個位元組的記憶體

3. [6] 請舉例說明什麼是變數的作用範圍 (scope of variables)?

   Sol:

   在程式裡定義一個變數以後，可以使用這個變數的程式區段就是這個變數的作用範圍，例如在一個函式裡宣告一個區域變數或是參數，從定義的地方開始一直到函式結束的右邊大括號為止都可以使用這個變數; 在函式外面宣告一個全域變數，從定義的地方開始一直到檔案結束任何程式碼都可以使用這個變數

```
int fun(int x) {
      double y;
…
…
}
```

4. [6] 請寫一個 **for** 迴圈，由小到大依序列印在 100 到 1000 之間是 3 或 7 的倍數的所有整數 (請以空格分隔每一個數字)

   Sol:

```
int i;
for (i=100; i<=1000; i++)
      if ((i%3==0)||(i%7==0))
            printf("%d ", i);
```

5. [6] 請寫一個函式 double *average*(int ndata, double data[]) 計算傳入的陣列中 ndata 個倍精準浮點數的平均值，並且回傳計算出來的數值

Sol:

```
double average(int ndata, double data[]) {
    int i;
    double sum=0;
    for (i=0; i<ndata; i++)
        sum += data[i];
    return sum / ndata;
}
```

6. [6] 請填寫下列空格，並且撰寫函式 *square* 將所傳入的三個參數對應的變數內容各自平方

```
int x=1,y=2,z=3;
square(___, ___, ___); // 希望修改 x, y, z 三個變數的數值為原本的平方
printf("x=%d y=%d z=%d", x, y, z); //螢幕輸出為 x=1 y=4 z=9
```

Sol:

```
square(&x, &y, &z);
----------------------------
void square(int *x, int *y, int *z)
{
    *x = *x * *x;
    *y = *y * *y;
    *z = *z * *z ;
}
```

7. [15] 請撰寫一個程式，讀入一長串(1~10000 個) 0/1 資料代表一個二進位的整數，例如 11011101101011101110010111101，請判斷這個數字是不是 3 的倍數 ($2^k = (3-1)^k = 3m + (-1)^k$)

Sol:

```
01 #include <stdio.h>
02 int main()
03 {
04     int sum=0, ndigits = 0;
05     char cbuf;
06     printf("The binary number: ");
07     while (scanf("%c", &cbuf) == 1)
08     {
09         printf("%c", cbuf);
10         sum += (ndigits%2==1 ? cbuf-'0' : '0'-cbuf);
11         ndigits++;
12     }
13     if (sum % 3 == 0)
14         printf(" is a multiple of 3.\n");
15     else
16         printf(" is NOT a multiple of 3.\n");
17     return 0;
18 }
```

題目沒有寫清楚該如何結束，用換列字元或是任何非 '0'/'1' 字元結束都可，例如
if ((cbuf!='0')&&(cbuf!='1'))
    break;

if (ndigits%2==1)
    sum += cbuf-'0';
else
    sum -= cbuf-'0';

8. [15] 下面程式片段由鍵盤讀入一個最多 20 位數的 16 進位的整數，轉換為 2 進位數字輸出，請撰寫 *convert* 函式以配合下列 main 函式運作，請加入需要的 include 敘述，執行範例如下右，請注意輸入除了數字 0 之外，第一個字元一定是'1'~'F'不會是'0'，輸出也是一樣:

```
01 #include <stdio.h>
02 void convert(char hex[], char binary[]);
03 int main() {
04     char hex[21];
05     char binary[81];
06     printf("請輸入 16 進位整數: ");
07     scanf("%s", hex);
08     convert(hex, binary);
09     printf("2 進位表示為: %s\n", binary);
10     return 0;
11 }
```

```
請輸入 16 進位整數: 0
2 進位表示為: 0

請輸入 16 進位整數: 234
2 進位表示為: 1000110100

請輸入 16 進位整數: 1234AB01
2 進位表示為: 10010001101001010101100000001
```

Sol: 想要一次就做好的寫法

```
01 void convert(char hex[], char binary[])
02 {
03      int i, j, ibinary, len, value, lenHex = strlen(hex);
04      for (i=0, ibinary=0; i<lenHex; i++)
05      {
06          if ((hex[i]>='0')&&(hex[i]<'9'))
07              value = hex[i] - '0';
08          else
09              value = hex[i] - 'A' + 10;
10          if (i==0)
11              for (len=1,j=value; j>1; j/=2) len++;
12          else
13              len = 4;
14          for (j=0; j<len; j++, value/=2)
15              binary[ibinary+len-1-j] = value%2 + '0';
16          ibinary += len;
17      }
18      binary[ibinary] = 0;
19 }
```

比較模組化的寫法

```
01 void convert(char hex[], char binary[])
02 {
03      int i, j, ibinary, value, lenHex = strlen(hex);
04      for (i=lenHex-1, ibinary=0; i>=0; i--)
05      {
06          if ((hex[i]>='0')&&(hex[i]<'9'))
07              value = hex[i] - '0';
08          else
09              value = hex[i] - 'A' + 10;
10          printf("value=%d\n", value);
11          for (j=0; j<4; j++, value/=2)
12              binary[ibinary++] = value%2 + '0';
13      }
14      while ((ibinary>1)&&(binary[ibinary-1]=='0'))
15          ibinary--;
16      binary[ibinary] = 0;
17      strrev(binary);
18 }
```

9. [15] 請撰寫一個程式，由鍵盤讀入一正整數 n，程式列印 n 列資料如下:

```
請輸入一個整數: 5
#####****#****
*###*****##***
**###*****###*
***###*****###*
****#****#####
```

```
請輸入一個整數: 6
######****#*****
*#####****##****
**####****###***
***###*****####**
****##*****#####*
*****#****######
```

Sol:

```
01 #include <stdio.h>
02
03 int main()
04 {
05      int i, j, n;
06      scanf("%d", &n);
07      for (i=0; i<n; i++)
08      {
09          for (j=0; j<i; j++)
10              printf("*");
11          for (j=0; j<n-i; j++)
12              printf("#");
13          for (j=0; j<4; j++)
14              printf("*");
15          for (j=0; j<i+1; j++)
16              printf("#");
17          for (j=0; j<n-i-1; j++)
18              printf("*");
19          printf("\n");
20      }
21      return 0;
22 }
```

10. [25] 假設有三個資料檔案 "data1.txt", "data2.txt", 及 "data3.txt", 每個資料檔案內有多筆(至少有一筆)由小而大按順序存放的整數資料 (資料的數值範圍在 0~99999 之間), 每個資料檔案內資料筆數不見得相同, 請寫一個程式讀取這三個檔案的資料內容, 在螢幕上由小至大輸出所有的資料:

data1.txt
```
9
11
22
56
…
```

data2.txt
```
22
25
44
47
58
…
```

data3.txt
```
7
8
11
24
…
```

螢幕輸出
```
7
8
9
11
11
22
22
24
25
44
47
56
58
…
```

Sol:

```
01 #include <stdio.h>
02
03 int main()
04 {
05      FILE *fp1, *fp2, *fp3;
06      int buf1, buf2, buf3, nitem1, nitem2, nitem3;
07      int i=0, status=0;
08
09      fp1 = fopen("data1.txt", "r");
10      fp2 = fopen("data2.txt", "r");
```

```
11      fp3 = fopen("data3.txt", "r");
12      nitem1 = fscanf(fp1, "%d", &buf1);
13      nitem2 = fscanf(fp2, "%d", &buf2);
14      nitem3 = fscanf(fp3, "%d", &buf3);
15
16      while ((nitem1==1)||( nitem2==1)||(nitem3==1)) {
17            if ((nitem1==1) &&
18               (((nitem2==0)||(buf1 <= buf2)) &&
19                ((nitem3==0)||(buf1 <= buf3)))) {
20                  printf("%d\n", buf1);
21                  nitem1 = fscanf(fp1, "%d", &buf1);
22            }
23            else if ((nitem2==1) &&
24                     (((nitem1==0)||(buf2 <= buf1)) &&
25                      ((nitem3==0)||(buf2 <= buf3)))) {
26                  printf("%d\n", buf2);
27                  nitem2 = fscanf(fp2, "%d", &buf2);
28            }
29            else if ((nitem3==1) &&
30                  (((nitem1==0)||(buf3 <= buf1)) &&
31                   ((nitem2==0)||(buf3 <= buf2)))) {
32                  printf("%d\n", buf3);
33                  nitem3 = fscanf(fp3, "%d", &buf3);
34            }
35      }
36
37      fclose(fp1), fclose(fp2), fclose(fp3);
38      return 0;
39 }
```

11. 有一個整數陣列 data 裡有 n 個數字，這些數字的範圍在 1~n-1 之間, $1 \le n \le 500$，其中只有一個數字出現多次，在下面限制下請分別寫出函式 int findDuplicate(int n, int data[])，在不修改原來陣列的情況下，找到並且回傳這個重複出現的數字

a. [5] 不可以使用額外的陣列，但是允許程式執行 $c\,n^2$ 次比對，其中 c 是常數

Sol: 不可以修改原來陣列，其實就是不希望你去排序 (排序的演算法運算量在 $O(n \log_2 n)$到 $O(n^2)$)

```
01 int findDuplicate(int n, int data[])
02 {
03      int i, j;
04      for (i=0; i<n; i++)
05            for (j=i+1; j<n; j++)
06                  if (data[i]==data[j]) return data[i];
07      return -1;
08 }
```

b. [5] 可以使用一個額外的 n 個元素的陣列，只允許程式執行 $c\,n$ 次運算，其中 c 是常數

Sol: 不可以修改原來陣列，雖然可以使用一個額外的 n 個元素的陣列，但是因為限制運算的次數正比於 n，所以還是排除了排序 (排序的演算法運算量次數正比於 $n \log_2 n$ 甚至 n^2)

```
01 int findDuplicate(int n, int data[])
02 {
03      int i, count[500] = {0}; // 陣列雖然定義 500 個元素，但是只會用到 n 個元素
04      for (i=0; i<n; i++)
05            if (count[data[i]] == 1)
06                  return data[i];
07            else
08                  count[data[i]] = 1;
09      return -1;
10 }
```

c. [15] 不可以使用額外的陣列，只允許程式執行 $c\,n \log_2 n$ 次運算, $\log_2$ 以 2 為底, c 是常數

Sol: 原則上 $\log_2 n$ 的演算法會是 divide-and-conquer 類型的方法，在這裡有兩種可能性，一是對於 n+1 個元素的陣列進行二分搜尋，但是問題在於針對當中的元素，例如 data[n/2]而言，如果去比對所有其他元素，發現沒有重複，就只能知道這個元素沒有重複，左半(data[0]~data[n/2-1])和右半(data[n/2+1]~data[n-1])的元素其實都還是不知道有沒有重複；第二種可能性是對於 n 個可能的數值 1~n 進行二分搜尋，對於中間的數值 n/2 來說，有沒有辦法確定 1~n/2 或是 n/2+1~n 之間包含那個重複的數字？以這題來說因為只有一個數字重複，所以可以拿 n/2 和所有數字比較，計算小於等於 n/2 元素的個數，如果是 n/2 個，也就是說 1~n/2 之間不包含那個重複的數字，反之則包含，於是需要搜尋的數字立刻減半，程式如下：

```
01 int findDuplicate(int n, int data[])
02 {
03      int max, min, medium, i, count;
04      max = n-1, min = 1;
05      while (max>min)
06      {
```

```
07              medium = (max + min) / 2;
08              for (i=0, count=0; i<n; i++)
09                      if (data[i]<=medium)
10                              count++;
11              if (count>medium)
12                      max = medium;
13              else // count==medium
14                      min = medium + 1;
15      }
16      return min;
17 }
```