1131 NTOUCSE 程式設計 1C 期中考參考答案

姓名：_____ 系級：_____ 學號：_____ 113/10/29 (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. 請回答下列程式問題：

(a) [5] 請寫一小段程式讀入如下圖的資料，其中整數是十進位 10～15 位數的整數，相加並且在 20 格的空間裡靠右對齊列印出來，下圖中 □ 代表空格，請使用適當的資料型態定義兩個變數 x, y，運用 scanf 讀入資料，用 printf 輸出兩數的和?

```
123456789012 □□+□□87654321
□□□□□□□□123544443333
```

Sol:

```
01 long long x, y;
02 scanf("%lld□+%lld", &x, &y);
03 printf("%20lld", x+y);
```

(b) [5] 下列程式希望用 printf() 完成如下圖的輸出格式，其中前面4位數是16進位，需要補足4位數不能有空格，接下來是逗點，還有整數部份 6 位數、小數部份 3 位數的十進位浮點數，圖中 □ 代表空格，請問格式轉換命令該如何指定?

```
003f,□□□□□1.234
```

```
01 int x;
02 double y;
03 scanf("%d%lf", &x, &y); // 讀取十進位 0~65535 的整數以及 0～999999 之間的十進位實數
04 printf("_____", x, y);
```

Sol:

04 printf("**%04x,%10.3f**", x, y);

(c) [10] 輸入資料如下圖，下列迴圈想要一次讀入一個十進位數字並且加總列印，請問使用 scanf() 時該怎麼做? 使用 getchar() 時該怎麼做?

```
1234
```

```
01 int i, x, sum;
02 for (sum=i=0; i<4; i++) {
03       _____
04       sum += _____
05 }
06 printf("sum is %d\n", sum); // 輸出 10
```

Sol:

```
03       scanf("%1d", &x);
04       sum += x ;
-------------------------------------------
03       x = getchar();
```

04 sum += **(x– '0') ;**

(d) [5] 下列程式想要一個字元一個字元讀取輸入串流裡的資料，可以正確編譯，但是執行的時候一直提早結束沒有辦法把所有資料讀完，請問發生了什麼狀況? 該怎麼修改?

```
01 char c;
02 while ((c=getchar())!=EOF) {
03       …
04 }
```

Sol:

replace **char** c; with **int** c;

This issue occurs due to the type of the variable c, it should be int c; such that it can hold the returned value of getchar(), which returns a value in the range 0~255 to represent the value of a byte and a value -1 (which is 0xFFFFFFFF) to denote the end of the input stream. If you use the type **int**, the normal input values in the range 0~255 will be converted to positive values 0x00000000~0x000000FF, while the end of file will be marked by 0xFFFFFFFF(EOF or -1). If you use the type **char**, in either cases the variable c will be equal to 0xFF such that the input byte 0xFF is the same as EOF.

(e) [5] 下列程式想要讀取下圖中兩列的資料，第一列是一個整數，第二列有一個字元，但是那個字元一直讀不到，請問是發生了什麼問題，運用 scanf() 函式該如何解決這個問題?

```
01 int x;
02 char ch;
03 scanf("%d", &x);
04 scanf("%c", &ch);
```

```
1234
y
```

Sol:

line 04 the variable ch will contains the invisible newline character '\n' after the digit '4' in the input stream. In order for this scanf() to work correctly, an additional space □ character as a command to scanf() should be inserted to the format string before %c as **scanf("□%c", &ch);** This command will skip all white spaces (space, \t, \n, \r) in the input stream and read the first non-white space character into the variable ch.

(f) [5] 請問下列程式為什麼很容易在執行的時候造成記憶體的錯誤存取? 甚至造成系統的漏洞遭受駭客的攻擊，要繼續使用 scanf() 讀取資料的話，最簡單的防止方法是什麼?

```
01 char buf[10];
02 scanf("%s", buf);
```

Sol:

It is very easy for a benign user to cause this code terminated with illegal memory access error by typing a string longer than 9 characters (scanf() always supplied an additional string terminator '\0' byte.) without any white space in it, let alone a hacker with malicious purpose. The input string will be filled to the character array buf[10], which consists of 10 contiguous bytes in the memory. If the input string is longer than 9 characters, scanf() will overwrite this memory area, whatever contents stored after this segment of memory will be destroyed. Some of them might be variables used by this program and overwriting them will cause unpredicted logic errors. Some of them might be used by other program in the system and cause the program to break down immediately. Some of them might disrupt the system mechanism and cause the execution of codes in the supplied data by the malicious hacker and thus the yield of system control to the hacker. In order to protect this scanf() from this type of problem, it is advised to supply a length limitation in the format string as **scanf("%9s", buf)** if the buffer has only 10 characters so that scanf() will never overwrite the 10 character buffer.

2. 請根據題目的敘述完成下列程式片段之功能

(a) [6] 請運用整數的除法和求餘數運算完成下列程式片段，將一個十進位整數 x 的 32 位元二進位反序列印在螢幕上(最低位元最先印出)。

```
01 int x=13;
```

```
02 printf("decimal: %d => binary: ", x);
03 for (; x>0; _____)
04         printf("%d", _____);
05 printf("\n");
```

執行結果

```
decimal: 13 => binary: 1011
```

Sol:

```
03 for (; x>0;  x/=2  )
04         printf("%d",  x%2  );
```

(b) [6] 接上題，想用一個陣列把低位元先紀錄下來，列印時由最高位元先印出：

```
01 int i, b[32], x=13;
02 for (i=0; x>0; _____)
03         _____ = _____;
04 while (i-->0)
05         printf("%d", b[i]);
06 printf("\n");
```

執行結果

```
decimal: 13 => binary: 1101
```

Sol:

```
02 for (i=0; x>0;  x/=2  )
03         b[i++]  =  x%2  ;
```

(c) [6] 接上題，不想用陣列想寫一個遞迴函式使得列印時由最高位元先印出：

```
01 void printBinary(const int x) {
02         _____
03                 printBinary(_____);
04         _____
05 }
06 int x=13;
07 printBinary(x);
08 printf("\n");
```

執行結果

```
decimal: 13 => binary: 1101
```

Sol:

```
02         if (x>1)  或是  if (x/2>0)  或是  if (x/2==0) { printf("%d", x%2); return; }  或是
03                 printBinary(  x/2  );          if (x<2) { printf("%d", x%2); return; }
04         printf("%d", x%2) ;
```

(d) [6] 接上題，不想用陣列也不想用遞迴，卻要由最高位元先印出，下列程式先計算出一個二的次方數 $w=2^k$，滿足 2*w>x>=w，再運用 w 來計算 x 的最高位元並且控制迴圈由 x 的最高位元開始列印：

```
01 int w, x2, x=13;
02 printf("decimal %d ==> binary ", x);
03 for (w=1,x2=x; x2_____1; x2/=2)
04         w*= _____;
05 for (; w>0; x%=w,w/=2)
06         printf("%d", _____);
07 printf("\n");
```

執行結果

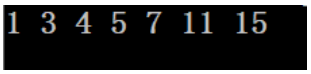
```
decimal: 13 => binary: 1101
```

Sol:

```
03 for (w=1,x2=x; x2  >  1; x2/=2)
04         w*=  2  ;
```

```
05 for (; w>0; x%=w,w/=2)
06       printf("%d",___x/w___);
```

3. [10] 有一個整數陣列裡面存放了遞增的整數，例如 1,1,1,3,4,4,5,7,11,11,15，請完成下列的程式片段來刪除陣列中所有重複的資料，以上面範例來說完成以後陣列中的資料剩下 1,3,4,5,7,11,15， 10-11 列程式執行印出來如右圖




```
01 #include <stdio.h>
02 int main() {
03       int i,j;
04       int x[]={1,1,1,3,4,4,5,7,11,11,15};          //運用 x[i] 來檢查陣列中每一資料，
05       for (j=0,i=1; i<sizeof(x)/___________; i++)  //運用 x[j] 來存放去除重複後的資料
06            if (_________________)                  //注意 x[j] 不是等於 x[i] 就是小於 x[i]
07                 x[_______] = x[i];
08       for (i=0; ___________; i++)
09            printf("%d%c", x[i],_________________); // 除了最後用換列字元之外，其它兩個數字間用空格
10       return 0;
11 }
```

Sol:

```
05       for (j=0,i=1; i<sizeof(x)/__sizeof(int)__; i++)
06            if (__x[i]__>__x[j]__)
07                 x[__++j__] = x[i];
08       for (i=0; __i__<=__j__; i++)
09            printf("%d%c", x[i],__i<j__?__'□'__:__'\n'__);
```

4. [20] 一個 DNA 片段的定序是 ATCG 四種鹼基組合的序列，我們要將許多的 DNA 序列依照它的逆序數來排順序，因此需要寫一個函式計算一個 DNA 序列的逆序數。一個 DNA 序列例如 “ATCGA” 的逆序數就是這個序列中逆序對 TC, TG, TA, CA, GA 的數量 5，英文字母的順序是 ABCDE…XYZ，如果一對字母的順序 (例如 TC) 和英文字母的順序相反就是一個逆序對。一般計算一個長度 n 的序列的逆序數最基本的方法就是用兩層迴圈把所有 C(n,2) 種組合的順序都檢查過，但是這樣的方法對於大量的 DNA 序列來說太花時間了，而且 DNA 序列裡只有 ATCG 四種鹼基，所以有一個簡單的方法從一個 DNA 序列的尾巴開始往前看，用三個整數變數 a, ac, acg 來紀錄目前出現的 A, C, G 的數量，其中 a 紀錄 A 出現的數量，ac 紀錄 A 和 C 出現的數量，acg 紀錄 A, C, 和 G 出現的數量，當看到一個字母 T 時，逆序數就是變數 acg 紀錄的數值，看到一個字母 G 時，逆序數就是變數 ac 紀錄的數值，看到一個字母 C 時，逆序數就是變數 a 紀錄的數值，請完成下列函式來計算一個 DNA 序列的逆序數：

```
01 #include <stdio.h>
02 int inversionNumber(char DNA[]) {
03       int inv, a, ac, acg, n, i;
04       n=0; while (____________!=0) n++;   // 計算 DNA[] 陣列裡面字元的個數
05       for (_____________________,i=n-1; i>=0; i--) { // 初始化 inv, a, ac, acg 變數
06            switch (_______) {
07                 case 'T':
08                      inv += _______;
09                      break;
10                 case 'G':
11                      inv += _______;
12                      ________++;
13                      break;
14                 case 'C':
15                      inv += _______;
16                      _________++, __________++;
```

```
17                      break;
18                  case 'A':
19                      _________++, _________++, _________++;
20              }
21          }
22          return _________;
23  }
24  int main() {
25          char DNA[] = "ATCGA";
26          printf("inversion number=%d\n", inversionNumber(DNA));
27          return 0;
28  }
```

Sol:

```
04      n=0; while (___DNA[n]___ !=0) n++;
05      for (__inv=a=ac=acg=0__,i=n-1; i>=0; i--) {
06          switch (__DNA[i]__) {
07              case 'T':
08                  inv +=___acg___;
09                  break;
10              case 'G':
11                  inv +=___ac___;
12                  ___acg___++;
13                  break;
14              case 'C':
15                  inv +=___a___;
16                  ___ac___++,___acg___++;
17                  break;
18              case 'A':
19                  ___a___++,___ac___++,___acg___++;
20          }
21      }
22      return ___inv___;
```

5. [11] 請完成下列迴圈來合併兩個已經由小到大排序的陣列 data1[], data2[] 到 data[]，資料仍然 j 維持由小到大排序

```
01 #include <stdio.h>
02 int main() {
03          int data1[]={6, 9, 11, 12, 15, 18};
04          int data2[]={1, 2, 3, 7, 14, 16, 20};
05          int data[100];
06          int i, idx1, idx2, idx, len1=6, len2=7, len;
07          idx = idx1 = idx2 = 0;
08          len = len1+len2;
09          while (_________ < len) {
10              if (_________ < len1 && _________ < len2) {
11                  if (data1[idx1] < data2[idx2])
12                      _________ = _________;
13                  else
14                      _________ = _________;
15              }
16              else if (idx1 < len1)
17                  _________ = _________;
18              else
19                  _________ = _________;
20          }
21          printf("len=%d: ", len);
```

```
22          for (i=0; i<idx; i++)
23              printf("%d%c", data[i], " \n"[i==idx-1]);
24          return 0;
25  }
```

Sol:

```
09      while (   idx    < len) {
10          if (   idx1    < len1 &&   idx2    < len2) {
11              if (data1[idx1] < data2[idx2])
12                  data[idx++]  =  data1[idx1++] ;
13              else
14                  data[idx++]  =  data2[idx2++] ;
15          }
16          else if (idx1 < len1)
17              data[idx++]  =  data1[idx1++] ;
18          else
19              data[idx++]  =  data2[idx2++] ;
```