計算機概論 資訊一甲 一零四學年度 第二學期 期中考試
學號：　　姓名：Solutions

1. (25 points) 回答下列的問題：
   (a) 假設 s 是一個 C 程式語言的字串變數，其起始位址為 0X401230 和字串內容為 “abcd321real”，寫出 strpbrk(strpbrk(s, "3"), "abcde") 的回傳值 (return value)。
   (b) 假設 s 是一個 C 程式語言的字串變數，其起始位址為 0X32C620 和字串內容為 “abcd321real”，寫出 strspn(strpbrk(s, "c"), "abcdefghij") 的回傳值 (return value)。
   (c) 假設 s1 和 s2 是兩個 C 程式語言的字串變數，其起始位址分別為 0X285C40 和 0X285C48，且 s1 和 s2 的字串內容分別為 “abcde” 和 “123”。寫出 strcat(s1, s2) 執行之後 s1 和 s2 的字串內容。
   (d) 繼續 (c) 小題 strcat(s1, s2) 執行之後，寫出 strpbrk(s1, s2) 的回傳值 (return value)。
   (e) 假設 s1 和 s2 是兩個 C 程式語言的字串變數，其起始位址分別為 0X285C40 和 0X285C48。寫出 strcpy(s1, “012345678”); strcpy(s2, s1); 執行之後 s1 和 s2 的字串內容。
   (a) strpbrk(strpbrk(s, "3"), "abcde") 的回傳值為 0X401238.
   (b) strspn(strpbrk(s, "c"), "abcdefghij") 的回傳值為 2.
   (c) strcat(s1, s2) 執行之後，s1 的字串內容是 abcde123, s2的字串內容是 NULL (空字串)。
   (d) strpbrk(s1, s2) 的回傳值為 0 (null pointer).
   (e) strcpy(s1, “012345678”); strcpy(s2, s1); 的執行會進入無限迴圈 (infinite loop)，因此無法得到 s1 和 s2 的字串內容。

2. (15 points) 考慮下列 C 語言程式片段：

```
int a[100];
void foo(unsigned char b[160]) {
  static short c[50];
  float d[120];
  float *e;
  …;
  e = (float *) malloc(800);
  …;
  free(e); }
```

```
int main(void) {
  double f[80];
  …;
  foo();
  …;
  foo();
  …; }
```

   (a) 變數 a, b, c, d, e, f 陣列元素依照記憶體的特性，指出何者為全域變數 (global variable)、區域變數 (local variable)、靜態變數 (static variable)、動態變數 (dynamic variable)。
   (b) 當執行上列程式時，系統分配及釋放 a, b, c, d, e, f 陣列元素的各類記憶體空間的時間點為何?
   (c) 變數 a, b, c, d, e, f 的陣列元素各佔多少位元組?
   (a) 變數 a 是全域變數 (global variable)；變數 b, d, 和 f 是區域變數 (local variable)；變數 c 是靜態變數 (static variable); 變數 e 是動態變數 (dynamic variable)。
   (b) 變數 a 的全域變數和變數 c 的靜態變數是在程式開始執行時就分配記憶體空間，直到程式執行結束為止；變數 b 和 d 的區域變數是當副程式 foo() 被呼叫時才分配記憶體空間，當 foo() 執行結束時其記憶體才被釋放；變數 f 是宣告在主程式的區域變數，其記憶體空間的分配如同全域變數，從程式開始執行至結束為止；變數 e 是動態變數的記憶體，是當 foo() 執行到 malloc() 副程式時才分配記憶體空間，一直執行到 free(e) 時就被釋放。
   (c) 變數 a 的陣列元素佔用 400 位元組；變數 b 的陣列元素佔用 160 位元組；變數 c 的陣列元素佔用 100 位元組；變數 d 的陣列元素佔用 480 位元組；變數 e 的陣列元素佔用 800 位元組；變數 f 的陣列元素佔用 640 位元組。

3. (15 points) 考慮下列 C 程式語言的主程式片段：

```
int main(int argc, char *argv[ ]) {
  …
}
```

假設此程式執行檔的檔名為 mygoodfriends.exe, 且此程式執行時的指令為：

mygoodfriends john bill mary susan tom

(a) 此程式片段執行時 argc 的值是什麼?

(b) 下列 main() 中程式碼的輸出值是什麼?

```
printf("%d, %d, %d", strcmp(argv[0], "MyGoodFriends"),
        strcmp(argv[2] , "john"), strcmp(argv[4] , "susan"));
```

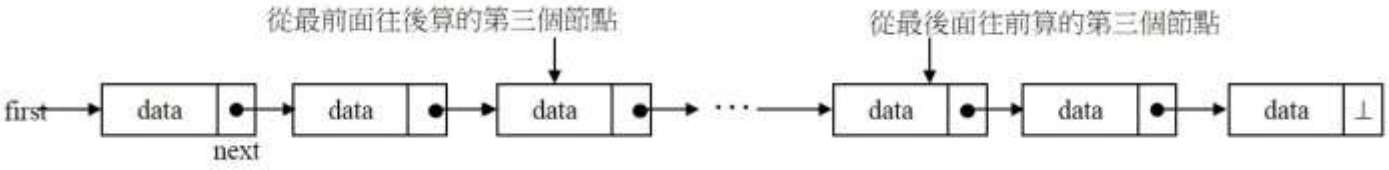
(c) 如果將 **int** main(**int** argc, **char** *argv[ ]) 寫成 **int** main(**int** num, **char** *str [ ]) 是否會引起程式編譯或執行錯誤，解釋你的原因。

(a) 6

(b) 1, -1, 0

(c) 不會引起程式編譯或執行錯誤。因為 argc 和 argv 只是主程式 main() 的參數名稱，只要使用 C 語言的合法名稱 (identifier)，並不會引起錯誤。

4. 下列圖示是一個單向連結表的示意圖，假設此單向連結表至少有三個節點。



(a) (5 points) 假設 data 是一個整數，寫出此單向連結表節點的型態定義。

(b) (10 points) 寫一個程式片段以刪除從最前面往後算的第三個節點，宣告必要的變數。

(c) (10 points) 寫一個程式片段以刪除從最後面往前算的第三個節點，宣告必要的變數。(提示：考慮恰好三個節點和多於三個節點的情況)

(a) 單向連結表節點的型態定義如下：

```
typedef struct Node {
  int data;
  struct Node* next;
} node;
```

(b) 刪除前面算起第三個節點：

```
node *ptr1, *ptr2;
ptr1 = first->next; // 從最前面往後算的第二個節點
ptr2 = ptr1->next; //從最前面往後算的第三個節點
ptr1->next = ptr2->next; // 修改第二個節點的 next 指標
free(ptr2); // 刪除原連結表的第三個結點
```

(c) 刪除後面往回算起第三個節點：

```
node *ptr4, *ptr3, *ptr2, *ptr1;
ptr4 = NULL; // 從後面往回算第四個節點；當 ptr4 為空時，只考慮最前面三個節點
ptr3 = first; // 從後面往回算第三個節點
ptr2 = ptr3 ->next; // 從後面往回算第二個節點
ptr1 = ptr2->next; // 從後面往回算第一個節點
while (ptr1 ->next!= NULL) { // 當後面還有節點時，尋找最後一個節點
  ptr4 = ptr3;
  prt3 = ptr2;
```

```
    ptr2 = ptr1;
    ptr1 = ptr1->next;
  }
  if (ptr4==NULL) { // 連結表只有三個節點
    first = ptr2; // 將最後第二個節點設為單向連結表的第一個節點
    free(ptr3); // 刪除原來最後的第三個節點
  }
  else {
    ptr4->next = ptr2; // 修改最後第四個節點的 next 指標
    free(ptr3); // 刪除原來最後的第三個節點
  }
```

5. (20 points) 假設 **unsigned long** flength(**char** *fname); 是 C 語言的一個副程式宣告 (function declaration)，參數 fname 代表一個檔案名稱的字串， flength 計算並回傳 fname 檔案的長度。寫出 flength 的程式碼。(提示：使用 fopen(), fseek(), ftell(), fclose() 等副程式)

```
unsigned long flength(char *fname) {
  unsigned long leng;
  FILE *fptr;
  fptr =  fopen(fname,"r");
  fseek(fptr, 0, SEEK_END);
  leng = ftell(fptr);
  fclose(fptr);
  return leng;
}
```