1051 NTOUCSE 程式設計 C 期末考參考答案

姓名：_____ 系級：_____ 學號：_____ 106/01/11(三)

考試時間：14:30 – 16:30

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、或是其他參考資料
2. 可以使用鉛筆，可以在題目卷上直接回答，但是**請在答案卷上註明某題號的答案在題目卷上**
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**

1. [5] 請在右側程式空格處填入程式，使得程式的輸出如下圖所示

```
    *
   *#*
  *#*#*
 *#*#*#*
*#*#*#*#*
```

```
#include <stdio.h>
int main() {
    int i, j, k=4, m=1;
    for (i=1; i<=5; i++,k--,m+=2) {
        for (j=1; j<=k; j++) printf(" ");
        for (j=1; j<=m; j++)
            if (j%2==1)
                printf("*");
            else
                printf("#");
            // printf("%c", j%2 ? '*' : '#');
        printf("\n");
    }
    return 0;
}
```

2. [5] 請改寫下列 switch 敘述，以 if-else 語法得到完全相同的效果

```
switch (x) {
    case 1:
        y = 'a';
        break;
    case 2: case 3:
        y = 'b';
        break;
    default:
        y = 'c';
}
```

if (x==1) y = 'a';

else if (x==2||x==3) y = 'b';

else y = 'c';

3. [5] 請問右側 func() 函式執行後列印出什麼？

(1,4)

```
void func() {
    char t, item[] = {'2', '8', '3', '1', '9'};
    int a, b, c, count = 5;
    for (a=0; a<count-1; a++) {
        c = a;
        t = item[a];
        for (b=a+1; b<count; b++) {
            if (item[b]<t) {
                c = b;
                t = item[b];
            }
            if ((a==2) && (b==3))
                printf("(%c,%c)\n", t, t+c);
        }
    }
}
```

4. [5] 請問下圖左程式執行後列印出什麼? **21**

```
#include <stdio.h>
int func2(int a[], int n) {
    if (n>=0)
        return func2(a,n-1) + a[n];
    else
        return 6;
}
int func1(int n) {
    int a[] = {5,4,3,2,1};
    return func2(a,n);
}
int main() {
    printf("%d", func1(4));
    return 0;
}
```

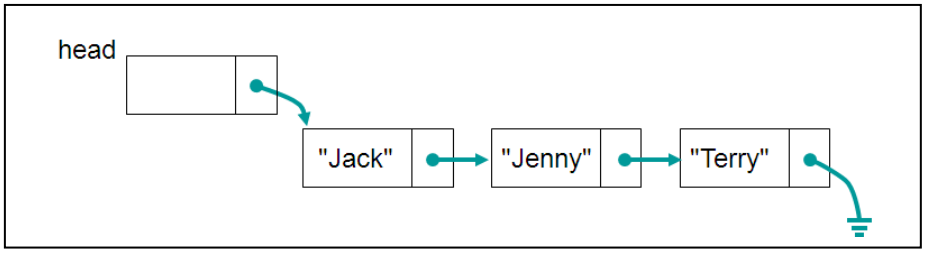
```
#include <stdio.h>
int func(int a) {
    if (a < 2)
        return 1;
    else
        return func(a-2) + func(a-3);
}

int main() {
    printf("%d\n", func(8));
    return 0;
}
```

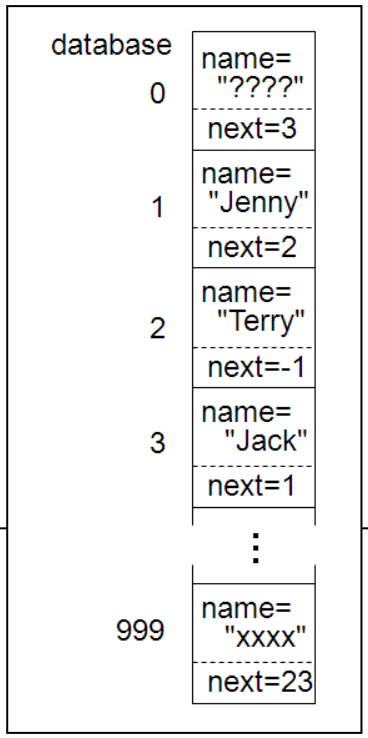
5. [5] 請問在上圖右空格中應該填入多少才會印出 9? **2**

6. [5] 請問下圖程式列印出來的數值是多少? **16**

```
#include <stdio.h>
int main() {
    int maze[5][5] =  {{1,1,1,1,1},
                       {1,0,1,0,1},
                       {1,1,0,0,1},
                       {1,0,0,1,1},
                       {1,1,1,1,1}};
    const int dir[4][2] = {{-1,0},{0,1},{1,0},{0,-1}};
    int i, j, d, count=0;
    for (i=1; i<=3; i++)
        for (j=1; j<=3; j++)
            if (maze[i][j]==0)
                for (d=0; d<4; d++)
                    if (maze[i+dir[d][0]][j+dir[d][1]]==1) count++;
    printf("%d\n", count);
    return 0;
}
```

7. 我們運用一個結構陣列 database 來儲存一串資料，例如右圖中 database[0] 的 next 欄位存放的資料為 3，代表這一串資料由陣列的第 3 個元素開始，database[2] 的 next 欄位存放 -1，代表放的是串列的最後一個元素；使用者自定的結構 struct Customer_t 定義如右下圖，概念上所代表的一串資料如下圖所示，設計這種方式來存放資料最主要的原因是資料可能需要頻繁地異動 (新增/刪除/更改順序/…)，如果只使用陣列來存放會造成整塊資料需要常常往前或是往後搬移，使得程式執行效率不佳。





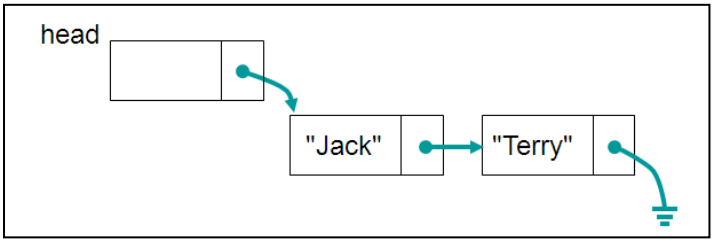
a) [3] 請運用 typedef 替這個結構 struct Customer_t 定義一個型態的別名 Customer

b) [3] 請定義一個可以放 1000 個元素的結構陣列 database

c) [6] 請完成右圖中的 findName 函式，由串列的第一個元素開始尋找指定姓名 tName 的節點，回傳那個節點的前一個節點在陣列中的位置，例如 int iPrev=findName(database, "Terry"); 執行後變數 iPrev 的數值為 1 請使用 strcmp 來比對，如果沒有找到請回傳 -1，請注意陣列中可能因為重複地新增與刪除顧客資料，造成陣列中很多資料已經不在串列中，如圖中的 database[999]

d) [3] 請完成右圖中的 removeNext 函式，將串列中指定元素的下一個元素移除，例如在上例中如果呼叫 removeNext(database, 3)會得到下面示意圖中的串列，其中 "Jenny" 那個顧客已經被刪除掉了，請注意由串列中刪除掉的元素仍然讓它在陣列中，不需要清除 (一般情況下會放進另外一個串列代表這些空間是可以使用的，不過這裡暫時不考量這一部份)

```
struct Customer_t {
        char name[10];
        int next;
};
typedef struct Customer_t Customer;
Customer database[1000];
int findName(Customer db[], char tName[]) {
        int prev=0;
        while (db[prev].next != -1)
                if (strcmp(db[db[prev].next].name, tName)==0)
                        return prev;
        return -1;
}
int removeNext(Customer db[], int current) {
        int toBeRemoved = db[current].next;
        if (toBeRemoved != -1)
                db[current].next = db[toBeRemoved].next;
        return toBeRemoved;
}
```

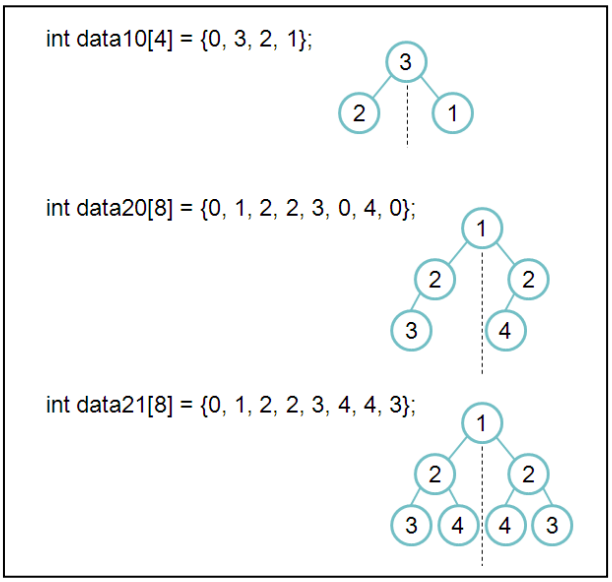


8. a) [5] 請問在左下方程式執行時輸入 23 時印出什麼? **8 18 23 23 found at data[9]\n**
   b) [5] 輸入 7 時程式印出什麼? **8 4 5 6 7 was not found!\n**
   c) [9] 請完成右下方 binarySearchRec 函式以**遞迴**方式實作左下的 binarySearch 函式

```
#include <stdio.h>
int binarySearch(int target, int data[], int left, int right) {
    int mid;
    while (left <= right) {
        mid = (left+right)/2;
        printf("%d ", data[mid]);
        if (target == data[mid])
            return mid;
        else if (target > data[mid])
            left = mid + 1;
        else
            right = mid - 1;
    }
    return -1;
}

int main() {
    int found, query, data[] = {2,3,4,5,6,8,13,16,18,23,24};
    while (1==scanf("%d", &query))
        if ((found=binarySearch(query, data, 0, 10))>=0)
            printf("%d found at data[%d]\n", query, found);
        else
            printf("%d was not found!\n", query);
    return 0;
}
```

```
int binarySearchRec(int target, int data[], int left, int right) {
    if (left > right) return -1;
    int mid = (left+right)/2;
    printf("%d ", data[mid]);
    if (target == data[mid])
        return mid;
    else if (target > data[mid])
        return binarySearchRec(target, data, mid+1, right);
    else
        return binarySearchRec(target, data, left, mid-1);
}
```

9. 右圖中左側陣列由第二個元素開始存放著右側的樹狀架構(看起來是倒過來的樹)的正整數資料，這個樹狀架構我們稱為二元樹，其中每一個節點下方最多連接兩個子節點，以陣列來存放時一層一層由上而下、由左而右依序放，陣列的第一個元素以及其它缺子節點的地方都存放 0；如果這個樹狀架構是自己的鏡像(對稱於圖形中間的虛線)，我們稱這樣的樹狀架構是對稱的(symmetric)，請在下圖空格中填入程式，以遞迴方式判斷陣列中的樹狀資料是不是對稱的?



   a) [6] 以 **data21** 陣列裡對稱的資料為例，可以看成是三個子序列 (1), (2, 2), (3, 4, 4, 3)，一個子序列裡面的資料如果由前面正的順序來看和由後面反的順序來看相同就稱為 Palindrome 序列，如果所有的子序列都是 Palindrome，就代表這個二元樹是對稱的。請撰寫 **int isPalindrome(int data[], int start, int end)** **遞迴**函式，判斷整數陣列 data[start]到 data[end]之間的數字是不是 Palindrome? 如果是的話回傳 1，否則回傳 0

```
int isPalindrome(int data[], int start, int end) {
    if (start>=end) return 1;
    if (data[start]!=data[end])
        return 0;
    else
        return isPalindrome(data, start+1, end-1);
}
```

或是

```
int isPalindrome(int data[], int start, int end) {
    return (start>=end) ||
        ((data[start]==data[end]) && isPalindrome(data, start+1, end-1));
}
```

b）[6] 請運用題a)的**isPalindrome(int[], int, int)**函式撰寫 **int isSymmetric(int data[], int depth)** **遞迴**函式，判斷深度為 depth 的二元樹狀架構的資料是不是對稱的？如果是的話回傳 1，否則回傳 0，上圖中 **data10** 陣列對應的樹的深度 depth 是 2, **data20** 和 **data21** 陣列對應的樹的深度 depth 都是 3。

```
int isSymmetric(int data[], int depth) {
    if (depth<=1) return 1;
    if (!isPalindrome(data, 1<<(depth-1), (1<<depth)-1))
        return 0;
    return isSymmetric(data, depth-1);
}
```

或是

```
int isSymmetric(int data[], int depth) {
    return (depth<=1) ||
        (isPalindrome(data, 1<<(depth-1), (1<<depth)-1)) && isSymmetric(data, depth-1));
}
```

c）[3] 請運用字元的指標陣列以及題 b)的 **isSymmetric** 函式完成右側的 main 函式，判斷並且列印陣列 **data20** 裡面的二元樹是否是對稱的

```
int main() {
    int data20[] = {0, 1, 2, 2, 3, 0, 4, 0};
    char *msg[] = {"is not symmetric", "is symmetric"};
    printf("data20 is %s\n", msg[isSymmetric(data20, 3)]);
    return 0;
}
```

10. 有一個人帶著一匹狼、一頭羊和一棵甘藍菜打算過河到對岸去。他的船很小，每次除了他自己之外最多只能載一個東西: 也就是船上只允許 (人), (人,狼), (人,羊), 或是(人,甘藍菜) 這四種組合，當然人如果不在船上的話船哪裡都去不了。現在麻煩的事情來了，如果他先載了甘藍菜過去，狼就會趁機把羊吃掉，如果他沒在旁邊看著羊，羊也會趁他離開時把甘藍菜吃掉，請寫一個**遞迴**程式找出這個問題有哪些解 (注意只要運送順序不完全一樣就算不同的解，但是在過程中相同的狀態不可以重複出現，例如一開始的時候把羊載到對岸再載回來，所以在過程中人狼羊菜同時在河岸這一邊出現第二次就不算是新的解)，雖然有很多方式可以撰寫這個程式，但是在這個測試裡請依照各個子題的說明在下列限制下完成這個程式:

a) [6] 解這個問題的時候我們需要想辦法在程式中表示出目前的狀態，我們可以用 0 代表河岸這一邊，用 1 代表河的對岸，然後把人狼羊菜在那一邊分別標示出來，例如 0 0 0 0 代表起始狀態，四者都在河岸這一邊, 1 0 1 0 代表人和羊到對岸去了，如果這是連續的兩個狀態，也就代表人載著羊到對岸去了，河的這一側留下狼和菜，請定義一個四個元素的一維整數陣列 currentState 來表示目前的狀態，因為我們需要檢查有沒有一些動作會導致先前的狀態重複出現，而且所有可能的狀態只有 2^4 種，請定義一個 16 個元素的一維整數陣列 visitedStates 來存放出現過的狀態, **1** 代表過程中已經出現的狀態, **0** 代表合法但是還沒有出現過的狀態，請分析一下哪些狀態是不允許出現的，把這些狀態初始為 **2**; 請定義一個二維的整數陣列 stepHistory 來紀錄過程中狀態出現的順序，因為可能出現的狀態有上限，又不可以重複，所以不需要定義太大的陣列，記錄在這個陣列裡的資料像是 0 0 0 0, 1 0 1 0, ….

```
int main() {
    int visitedStates[16] = {0,0,0,2, 0,0,2,2, 2,2,0,0, 2,0,0,0}; // 請定義可以快速查詢是否狀態
                                                                   //    已經出現過的陣列並且初始化
    int stepHistory[10][4] = {{0,0,0,0}}; // 請定義一個二維的整數陣列來紀錄狀態出現的順序
                                          //       步驟 0 請紀錄起始狀態 0 0 0 0
    int currentState[4]={0,0,0,0}; // 請定義目前狀態陣列變數並且初始化
    findSolution(currentState, 1, stepHistory, visitedStates); // 遞迴呼叫, 由步驟 1 開始尋找解
    return 0;
}
```

b) [5] 每一個目前的狀態 currentState 是用四個元素的一維整數陣列來表示，但是出現過的狀態是用一個 16 個整數的一維陣列 visitedStates 表示，我們需要在狀態 1 1 1 0 時將 visitedStates[14] 標示為 1，因此需要一個簡單的函式將 currentState 陣列中概念上二進位的表示方法 1 1 1 0 轉換為對應的十進位整數 14，請撰寫一個轉換函式 **int toDecimal(int state[])**

```
int toDecimal(int state[]) {
    int i, sum=0;
    for (i=0; i<4; i++) sum = sum * 2 + state[i];
    return sum;
}
```

c) [10] 請撰寫**遞迴**函式 **findSolution(int cState[], int step, int stepHistory[][4], int visitedStates[])**, 參數 cState 是目前的狀態, step 代表目前是第幾個步驟, stepHistory[0]是 0 0 0 0, 接下來走的步驟要紀錄在這個陣列中, visitedStates 陣列可以快速查詢是否已經走過某一狀態或是某一狀態是否為允許的狀態

```
void findSolution(int cState[], int step, int stepHistory[][4], int visitedStates[]) {
    static int nsols=0; // 記錄總共有多少解的變數
    int i, j, k, nstate, nState[4]; // 下一個評估中的新步驟的狀態
    for (i=0; i<4; i++) { // 船上只允許 (人), (人,狼), (人,羊), (人,甘藍菜) 其中一種
        if (cState[0]!=cState[i]) continue; // 人需要和想要搬運的東西在同一側
        for (j=1; j<4; j++) nState[j] = cState[j]; // 初始化新的狀態
        nState[0] = 1 - cState[0];                  //  人一定要在船上
        if (i>0) nState[i] = 1 - cState[i];          // 要載運的東西要在船上
        for (j=0; j<4; j++) stepHistory[step][j] = nState[j];   // 紀錄新步驟的狀態
        nstate = toDecimal(nState);
        if (nstate==15) { // 達成目標
            printf("%d: ", ++nsols); // 列印解的個數
            for (j=0; j<=step; j++) { // 列印每一步驟狀態的改變狀況
                for (k=0; k<4; k++) printf("%1d", stepHistory[j][k]);
                printf(" ");
            }
            printf("\n");
        }
        else if (!visitedStates[nstate]) { // 檢查 nState 是否可以走
            visitedStates[nstate] = 1;   // 標示下一步走到第 nstate 個狀態
            findSolution(nState, step+1, stepHistory, visitedStates); // 剩下所有可能性
            visitedStates[nstate] = 0;   // 還原狀態紀錄, 尋找下一個可能性
        }
    }
}
```

列印出來的結果如右圖:

```
1: 0000 1010 0010 1110 0100 1101 0101 1111
2: 0000 1010 0010 1011 0001 1101 0101 1111
```