計算機概論與程式設計 – Take-home Final
2023 JAN

You have to turn in programs for each sub-question. (Total 120 points)
You have to write comments for important statements to explain how it works.

I. Text processing programs.

1-a) Write a word-count program that can read a paragraph (ending with a blank line) and compute its number of alphabet characters and number of words (ignore space and any other punctuations). (8pts)

This is a book.

output:
11 4

Ans:
這一題最主要的概念就是

1. 用 **ascii** 碼判斷是否為英文字。
   Ascii 碼 A~Z:65~90，a~z:97~122。

```
// calculate alphabet characters
for(int i=0;i<MAX_LEN;i++)
{
    int ah=arr1[i];      //alphabet to ASCII code
    if(ah>=65 && ah<=90) //A~Z ASCII code
    {
        alpha_num++;
    }
    else if(ah>=97&&ah<=122) //a~z ASCII code
    {
        alpha_num++;
    }
}
```

2. 用 空白 和 "**\n**" 進行字串分割，然後計算 **word** 數量。
   主要用到 **strtok** 套件進行操作。

```
// use blank(" ") to splite string
const char s[3] = " \n";
// calculate number of words
char *token;
token = strtok(arr1, s);
while(token!=NULL)
{
    //printf("t=%s wm=%d\n",token,word_nums);
    word_nums+=1;
    token = strtok(NULL, s);
}
```

1-b) Write a program that reads a paragraph and performs correct capitalization for each word. (8pts)

This is a bOOK.

output:

This Is A Book.

Ans:

這一題主要概念為:

1. 先全部轉成小寫。
   我先定義大小寫轉換

```
// 大->小寫轉換
char bigTosmall(char c){
    if (c >= 'A' && c <= 'Z')
    {
        c = c + 32;
    }
    return c;
}

// 小->大寫轉換
char smallTobig(char c){
    if (c >= 'a' && c <= 'z')
    {
        c = c - 32;
    }
    return c;
}
```

   然後全部轉小寫

```
for(int i=0;i<MAX_LEN;i++)
{
    // 1. 全部轉小寫
    arr1[i] = bigTosmall(arr1[i]);
}
```

2. 如果是第一個字元，那字元一定是大寫!
3. 如果字元遇到空白，那下一個字元一定是大寫!
4. 如果字元遇到"\n"，那下一個字元一定是大寫!

   然後根據條件 2~4 把小寫轉大寫!!

```
for(int i=0;i<MAX_LEN;i++){
    // 2. 開頭要大寫
    if(i==0){
        arr1[i] = smallTobig(arr1[i]);
    }
    // 3.其他空白後面要大寫
    else if(arr1[i]==' '){
        arr1[i+1] = smallTobig(arr1[i+1]);
    }
    // 4.換行後面要大寫
    else if(arr1[i]=='\n'){
        arr1[i+1] = smallTobig(arr1[i+1]);
    }
}
```

1-c) word replacement: Now we allow a new operations: **replace old new** to replace all occurrences of **old** words by **new** word. (12pts)

This book is a good book.

(blank)

replace book tiger

replace good bad

output:

This tiger is a bad tiger.

Ans:

這一題主要概念為:

1. 要創建一個對照字典，讓字元可以對照，然後取代，所以我創一個 2D 的 **array**，如下表:

Old pattern	New pattern
Book	tiger
Good	bad

```
// 1. 創建 replace 字典 比較字串!!
char* replace_dict[MAX_LEN][2];
int N=0;
```

2. 因為標點符號也要留下來，我也有創一個 **token array** 放切割好的字串(有標點符號)，例如:

This	book	is	a	good	Book.

```
// 2. 創建切割好的字串array
char* tokenArray[MAX_LEN];
int t=0;
```

3. 然後使用 空白 切割字串，並將字串放進字典和 **token array**。

```c
// 3. use blank(" ") and "\n" to splite string
const char s[3] = " \n";
char *token;
token = strtok(arr1, s);
while( token != NULL ){
    if (strcmp(token,"replace")==0){
        // 放進字典
        token = strtok(NULL, s);   // old pattern
        replace_dict[N][0]=token;
        token = strtok(NULL, s);   // new pattern
        replace_dict[N][1]=token;
        N++;
    }
    // 放進token array
    tokenArray[t] = token;
    t++;
    token = strtok(NULL, s);
}
```

4. **Replace** 的方法主要就是利用 **strstr** 套件找到要取代的字串的起始位址 **index**，然後將 **old pattern** 依照長度填入 **new pattern**。詳細步驟如下圖所述。

```c
char *str_replace (char *source, char *find,  char *rep){
  // old pattern的長度
  int find_L=strlen(find);
  // new pattern的長度
  int rep_L=strlen(rep);
  // 字串的長度
  int length=strlen(source)+1;
  // 定位偏移量
  int gap=0;
  // 建立字串，並複製文字
  char *result = (char*)malloc(sizeof(char) * length);
  strcpy(result, source);
  // 尚未被取代的字串
  char *former=source;
  // old pattern出現的起始位址指標
  char *location= strstr(former, find);
  // 漸進搜尋欲替換的文字
  while(location!=NULL){
      // 增加定位偏移量
      gap+=(location - former);
      // 將結束符號定在搜尋到的位址上
      result[gap]='\0';
      // 計算新的長度
      length+=(rep_L-find_L);
      // 變更記憶體空間
      result = (char*)realloc(result, length * sizeof(char));
      // 替換的文字串接在結果後面
      strcat(result, rep);
      // 更新定位偏移量
      gap+=rep_L;
      // 更新尚未被取代的字串的位址
      former=location+find_L;
      // 將尚未被取代的文字串接在結果後面
      strcat(result, former);
      // old pattern出現的起始位址指標
      location= strstr(former, find);
  }
```

然後將 **token array** 裡面的字串一個一個拿出來和字典比對並取代。

```
// 4. 取代字串!!
for(int j=0;j<N;j++){
    for(int i=0;i<t-N;i++){
        char* str2 = str_replace(tokenArray[i], replace_dict[j][0], replace_dict[j][1]);
        tokenArray[i] = str2;
    }
}
```

1-d) Write a program that reads a paragraph and compute the occurrences of each word. Order the words by frequency (by the word length if the same frequency). Print out the top 5 frequent words. (You should be able to ignore the upper case or lower case.) (15pts)

The red book is the most popular book in the market.

output:
the 3
book 2
popular 1
market 1
most 1

Ans:

概念上為

1. 先創一個字典，為 2D 矩陣。用來第一行放字，第二行放字數。

word	Word num
the	0
Book	0
Popular	0
most	0

2. 利用空白和標點符號先切割字串。
3. 如果 **word** 有在字典裡，那字典相對 **word num** 就加 **1**。
4. 如果 **word** 沒有在字典裡，那就在字典填入 **word**。
5. 最後依照大小順序排列 **word num**，印出前 **5** 多的字**&**字數。

II. Write a sort program that can organize a given series of scores (ending with 0) and output them in a descending order. For each program, you also have to show the execution time to compare your program efficiency.

2-a) Implement by an array: Read the series of scores into an array, use a **largest()** to find the largest number, then use a **swap()** function to swap two integers so that the entire array is sorted (as we did in the mid-term exam). Output the numbers to stdout.

Your program should consider the count of the score series is unknown. The max size you

can use for **malloc()** is 4KB each time. If not sufficient, you can double it each time. If you use **realloc** on size overflow, you should avoid dangling problem. (10pts)

But it costs lots of copy. You will get extra bonus if you use one **malloc()** as a chunk to store 1K numbers (never **free()**) and maintain another super pointer array for multiple chunks. You can implement an **index(i)** to get the exact the i^{th} number pointer.

2 14 5 90 2 40 0

output:

90 40 14 5 2 2

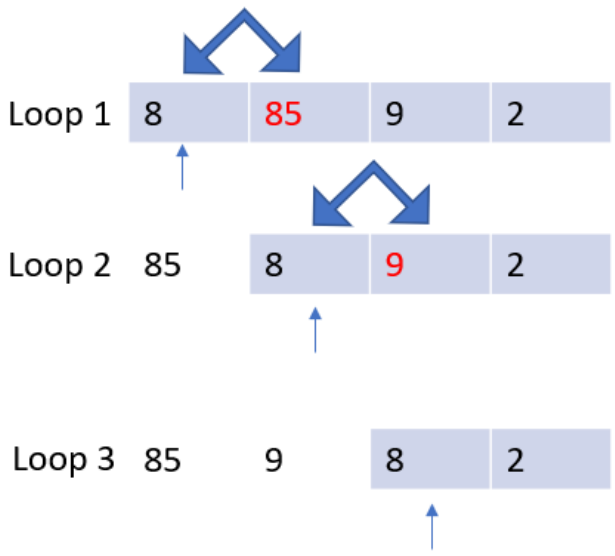
Ans:

主要概念為:

1. 使用之前的期中考試的 **largest**，回傳最大值的 **pointer**。

```c
int* largest(int* arr, int N)
{
    int i;
    int *max;
    max = arr;
    for (i = 1; i < N; i++) {
        if (*max < *(arr + i)) {
            max = (arr + i);
        }
    }
    return max;
}
```

2. 然後模擬 **bubble sort** 直接一個 **for** 迴圈每次都將最大的值，往前交換。步驟如下圖所述，注意:丟進去找最大值的 **array** 會越來越小**(N-i)!**



```
// bubble sort
for (int i=0; i<N; i++) {
    int *L_P = largest(&(arr[i]), N-i);  // find largest number and swap
    swap(L_P, &arr[i]);
}
```

3. 交換(**swap**)也是用 **pointer** 直接交換。

```
void swap(int* a, int* b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

2-b) Sort numbers in an array: You have to implement **add_num()** and **del_num()** functions to deal with the number array always in a descending order. Now when the program reads each number initially, you perform many **add_num()** operations until finding 0. In the input, we also allow two operations: **add *nn nn*** … and **del *nn nn*** … to insert and delete any number from the series. One operation appears in a line only. Again if you implement a chunk mechanism, it can save more execution time. (12pts)

```
2 14 5 90 2 40 0
add 86 50 1
del 40 5
add 55
```

```
output:
90 86 55 50 14 2 2 1
```

Ans:

主要概念為:

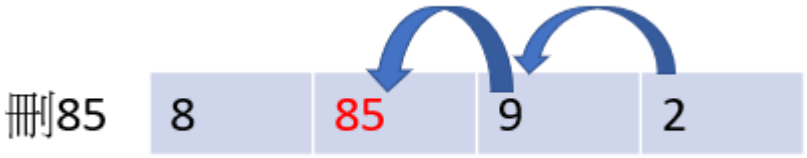
1. 創一個 **array** 放 **initial** 的數據值。

```
// 1. initial array
while(scanf("%d", &temp) && temp!=0) arr[N++]=temp;
```

2. 然後切割字串時，先判斷是否為"**add**"，如果是就將 **add** 後的數字，放進 **array** 裡面。所以我就沒有特別寫 **add_num** 的 **function** 了。

```c
if (strcmp(token, "add")==0){
    token = strtok(NULL, s);
    while( token != NULL ){
        arr[N]=atoi(token);     // 直接將字加進array
        N++;
        token = strtok(NULL, s);
    }
```

3. 如果是"**del**"，就去 **array** 裡面一個一個找，如果找出符合，就記錄下欲刪除的值的 **index**，並把它後面的值都往前移一格。




```c
int* del_num(int* arr, int del_n, int N)
{
    /*
    arr   is array
    del_n is delete number
    N     is array size
    */
    int break_index=0;
    for (int i = 0; i < N; i++)
    {
        if (arr[i]==del_n)
        {
            break_index=i;
            break;
        }
    }

    // from break_index 開始把資料往前移
    for (int i = break_index; i < N; i++)
    {
        arr[i]=arr[i+1];
    }

    return arr;
}
```

4. 排序部分其實就是使用 **bubble sort**，但是這次需要兩個 **for loop** 迴圈處理!!

```c
void bubblesort(int n, int* ptr)
{
    int i, j, t;
    for (i = 0; i < n; i++) {

        for (j = i + 1; j < n; j++) {

            if (*(ptr + j) < *(ptr + i)) {

                t = *(ptr + i);
                *(ptr + i) = *(ptr + j);
                *(ptr + j) = t;
            }
        }
    }
}
```

2-c) Implement by linked list: the same sort functionality as **1-b** program. We also support two operations: **add *nnn*** and **del *nnn*** to insert and delete any number from the series. (Hint: You have to use a double-pointer mechanism for **add_num()** and **del_num()** functions.)

In your final documentation, compare the execution time of 1-a, 1-b and 1-c (or chunk mechanism). Give your explanation. (15pts)

Ans:

主要概念如下:

1. 一樣創一個 **linked list** 的 **struct** 放 **initial** 的數據值。

```c
struct node
{
    int data;
    struct node *next;
};
```

2. 一樣切割字串，如果遇到"**add**"就將值放進 **link list** 尾部。

```c
void add_num(struct Node** head_ref, int new_data)
{
    /* 1. allocate node */
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));

    /* 2. put in the data  */
    new_node->data  = new_data;

    /* 3. Make next of new node as head */
    new_node->next = (*head_ref);

    /* 4. move the head to point to the new node */
    (*head_ref)    = new_node;
}
```

3. 如果遇到"**del**"就去 **link list** 裡面一個一個搜尋相對應值，並刪除。

```
void del_num(Node** head_ref, int key)
{
    Node* temp = *head_ref;
    Node* prev = NULL;

    if (temp != NULL && temp->data == key)
    {
        *head_ref = temp->next; // Changed head
        delete temp;            // free old head
        return;
    }


      else
    {
    while (temp != NULL && temp->data != key)
    {
        prev = temp;
        temp = temp->next;
    }
    if (temp == NULL)
        return;

    prev->next = temp->next;

    delete temp;
    }
}
```

I. Misc and bonus programs.

3-a) Write a program that can read a date format yyyy/mm/dd or ROC yyy/mm/dd (中華民國紀元). Then print it in a form of day name in the week, month name, date and year (AD). The input range is only between 2021/1/1~2023/12/31. Just ignore the lunar year. (10pts)

input
2023/1/9 ROC 112/1/8 ROC 111/12/31

output
January 9, 2023 (Monday) January 8, 2023 (Sunday) December 31, 2022 (Saturday)

Ans:

主要概念如下:

1. 如果開頭是'R'，就是 ROC yyyy/mm/dd 格式，用 ROC 專門的 token。

```c
void ROC_token(char* commend){
    int month = 0;
    int year  = 0;
    int day   = 0;

    char Week[20]={'\0'};
    const char s1[2] = " ";
    const char s2[2] = "/";
    char *token;

    token = strtok(commend, s1);

    token = strtok(NULL, s2);
    year  = atoi(token)+1911;

    token = strtok(NULL, s2);
    month = atoi(token);

    token = strtok(NULL, s2);
    day = atoi(token);

    print_month(month);
    printf("%d, ", day);
    printf("%d ", year);
    int w = week(year, month, day);
    print_week(w);
}
```

2. 如果是普通的就直接 token。

```c
void token(char* commend){
    int month = 0;
    int year  = 0;
    int day   = 0;

    char Week[20]={'\0'};
    const char s2[2] = "/";
    char *token;

    token = strtok(commend, s2);
    year  = atoi(token);

    token = strtok(NULL, s2);
    month = atoi(token);

    token = strtok(NULL, s2);
    day = atoi(token);

    print_month(month);
    printf("%d, ", day);
    printf("%d ", year);
    int w = week(year, month, day);
    print_week(w);
}
```

3. 月、星期幾就用 **`case switch`** 去做替換。

```c
void print_month(int month){
    switch (month){
    case 1:
        printf("January ");
        break;
    case 2:
        printf("February ");
        break;
    case 3:
        printf("March ");
        break;
    case 4:
        printf("April ");
        break;
    case 5:
        printf("May ");
        break;
    case 6:
        printf("June ");
        break;
    case 7:
        printf("July ");
        break;
    case 8:
        printf("Auguest ");
        break;
    case 9:
        printf("September ");
        break;
    case 10:
        printf("October ");
        break;
    case 11:
        printf("November ");
        break;
    case 12:
        printf("December ");
        break;
```

```c
void print_week(int w){
    switch (w)
    {
    case 1:
        printf("(Monday)\n");
        break;
    case 2:
        printf("(Tuesday)\n");
        break;
    case 3:
        printf("(Wednesday)\n");
        break;
    case 4:
        printf("(Thursday)\n");
        break;
    case 5:
        printf("(Friday)\n");
        break;
    case 6:
        printf("(Saturday)\n");
        break;
    case 0:
        printf("(Sunday)\n");
        break;
    default:
        break;
    }
}
```

4. 重點是 **week** 的計算是參考基姆拉爾森計算公式。可以用 **year/month/day** 找出 **week**。

```c
int week(int year, int month, int day){
    int y,y0,m,m0,d,d0,z,x;
    y = year;
    m = month;
    d = day;

    z=(14-m)/12;
    y0=y-z;
    x=y0+y0/4-y0/100+y0/400;
    m0=m+12*z-2;
    d0=(d+x+31*m0/12)%7;
    return d0;
}
```

3-b) A valid password has the following requirements. For given multiple lines that contain one word. Print out the validation of each password. (10pts)

a) Password must contain at least one upper case;
b) Password must contain at least two lower case letter letters and all have to be **strictly inside** (i.e. not as the first or the last character);
c) Password must contain at least one digit [0-9] and all digits have to be **strictly inside**;
d) Must contain at least one special character from the set { '@', '!', '%', '&', '*' };
e) Password must be at least 8 characters in length, but it can be longer.

input	output
BjdaA00C! U_R_kill@final0109 Ha!ha!ha!	yes no no

Ans:

主要概念如下:

a) 其實主要就是利用前面 **ASCII** 碼 **A~Z:65~90**，判斷大寫出現次數。

b) 其實主要就是利用前面 **ASCII** 碼 **a~z:97~122**，判斷小寫出現次數。只是要注意從 **index:1~N-2,**頭尾不算。

c) 其實主要判斷是否為 **int**。只是要注意從 **index:1~N-2,**頭尾不算。

d) 這其實也只是 **ASCII** 碼判斷，**'@'=64, '!'=33, '%'=37, '&'=38, '*'=42**。

e) 這其實只是算字元數，字元數要**>8**。

3-c) Write a program that can read a digit string [0~9] and also a few following patterns. Your job is to determine if the pattern is a substring of the given input string and also give the string position (starting from 0). (5pts)

input	output
20230109123456543210 12345 20320109 3210	Yes 8 No x No x

Ans:

主要概念如下:

1. 主要為比較子字串，那就可以參考 **q2** 字串取代的部分，使用套件 **strstr** 直接判斷是否有相同，然後 **output index**。

```c
char *str_replace (char *source, char *find,  char *rep){
    // old pattern的長度
    int find_L=strlen(find);
    // new pattern的長度
    int rep_L=strlen(rep);
    // 字串的長度
    int length=strlen(source)+1;
    // 定位偏移量
    int gap=0;
    // 建立字串，並複製文字
    char *result = (char*)malloc(sizeof(char) * length);
    strcpy(result, source);
    // 尚未被取代的字串
    char *former=source;
    // old pattern出現的起始位址指標
    char *location= strstr(former, find);
    // 漸進搜尋欲替換的文字
    while(location!=NULL){
        // 增加定位偏移量
        gap+=(location - former);
        // 將結束符號定在搜尋到的位址上
        result[gap]='\0';
        // 計算新的長度
        length+=(rep_L-find_L);
        // 變更記憶體空間
        result = (char*)realloc(result, length * sizeof(char));
        // 替換的文字串接在結果後面
        strcat(result, rep);
        // 更新定位偏移量
        gap+=rep_L;
        // 更新尚未被取代的字串的位址
        former=location+find_L;
        // 將尚未被取代的文字串接在結果後面
        strcat(result, former);
        // old pattern出現的起始位址指標
        location= strstr(former, find);
    }
```

3-d) The same function as 3-c. The decision becomes checking "**anagram of secret**" (a new number is created by rearranging every single digit found in the given patterns), the same as we did in past mid-term exam.(5pts)

input
2023010912345654321 12345 20320109 3210

output
Yes 8 14 Yes 0 Yes 2