

DS HW3

M56121041 黃啟維

程式解說

print_add_minus (char *filename, int arra_len, int arrb_len)

我將原本main裡面的print和fuction call它寫一個fuction包刮了以下幾個功能

1. file proccessing (將file裡的兩行讀成兩個arr) 但要給兩個arr的int各數
2. construct node list a和b
3. 呼叫plus和minus函數做兩poly的加減法 並將結果印出
4. 解決memory leak問題

```
1  #define MAXREAD 10
2  void print_add_minus(char *filename, int arra_len, int arrb_len){
3      FILE *in;
4      in = fopen(filename, "r");//只讀就好
5      if (!in) {
6          perror("Error opening file");
7          return;
8      }
9      int reada[MAXREAD] = {0}, readb[MAXREAD] = {0};
10     for (int i = 0; i < arra_len; i++) {
11         fscanf(in, "%d", &reada[i]);
12     }
13
14     // 讀取第二行到 arr2
15     for (int i = 0; i < arrb_len; i++) {
16         fscanf(in, "%d", &readb[i]);
17     }
18     fclose(in);
19     Node *A = construct(reada, arra_len);
20     Node *B = construct(readb, arrb_len);
21     printf("%s\n", filename);
22     printf("\n");
23
24     printf("A:");
25     print_poly(A);
26
27     printf("B:");
28     print_poly(B);
29     Node *sum = plus(A, B);
30     printf("A+B:");
31     print_poly(sum);
32     Node *diff = minus(A, B);
33     printf("A-B:");
34     print_poly(diff);
35     printf("\n");
36
37     free_poly(A);
38     free_poly(B);
39     free_poly(sum);
40     free_poly(diff);
41 }
```

所以main舊址會有檔名和arr int數

```

1  int main(){
2      // Node declare
3
4      // file read and construct two given polynomial
5
6      // A+B and A-B
7
8      //For print
9      print_add_minus("test1.txt", 6, 8);
10     print_add_minus("test2.txt", 8, 8);
11     print_add_minus("test3.txt", 8, 8);
12     print_add_minus("test4.txt", 8,8);
13     print_add_minus("test5.txt", 6,6);
14     return 0;
15 }

```

Node* construct(int arr[], int listsize)

我的construct是以ptp去實做 先創造一個*head並用 ** curr去指向它
 先用** curr去尋找新的node的expo在list中的大小位置
 找到後插入這個新的node 並且再次將** curr指向head
 即可達到降冪排列 並回傳head

```

1  Node* construct(int arr[], int listsize ){
2      if(listsize < 2 && arr == NULL)return NULL;
3      Node *head = new_node(arr[0], arr[1]);
4      for(int i = 2; i < listsize ; i += 2){
5          int coef = arr[i];
6          int expo = arr[i+1];
7          //創件一個ptp
8          Node **curr = &head;
9          //找新的node的linklist中的位置 *curr要存在且下一個expo不能比新的大
10         while (*curr != NULL && (*curr)->expo > expo){
11             //把**curr一到下一個node
12             curr = &((*curr)->next);
13         }
14         //如果下一個和新的相同次數的加在一起
15         if (*curr != NULL && (*curr)->expo == expo){
16             (*curr)->coef += coef;
17             //如果下一個項相加後係數是零就把它去掉
18             if ((*curr)->coef == 0){
19                 Node *temp = *curr;
20                 *curr = (*curr)->next;
21                 free(temp);
22             }
23         }
24         else {
25             //在下一個創一個新的node
26             Node *temp = new_node(coef, expo);
27             temp ->next = *curr;
28             *curr = temp;
29         }
30     }
31     return head;
32 }
33 }

```

Node* new_node

要malloc一個新的node

```

1  Node* new_node(int coef, int expo){
2      Node *node = malloc(sizeof(Node));
3      node->coef = coef;
4      node->expo = expo;
5      node->next = NULL;
6      return node;
7  }

```

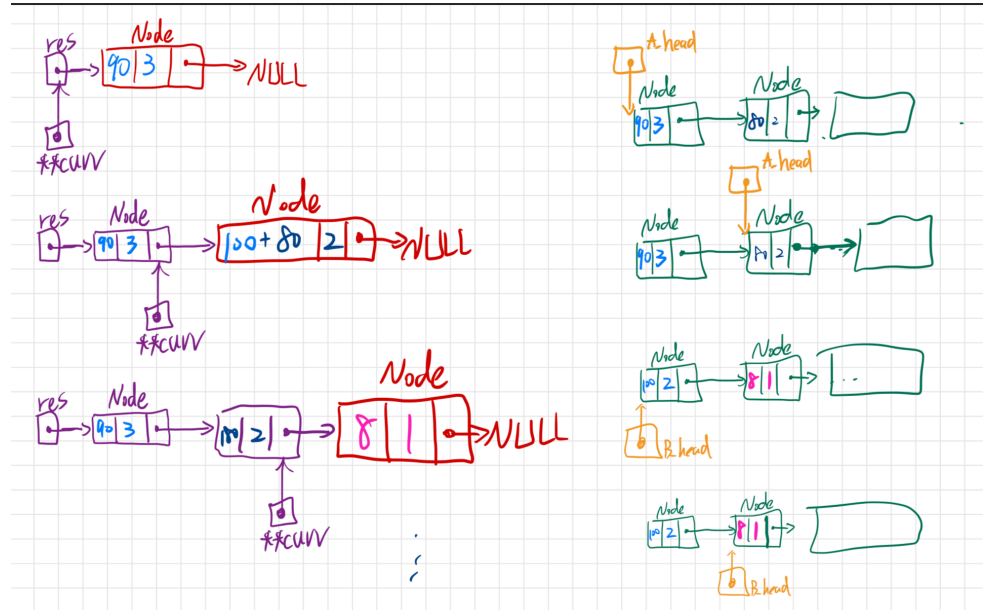
Node* plus(Node * A_head , Node* B_head)

比較兩個head的次數把大的複製進res list中

若A B次數相等則把兩個的係數相加放進res list中

直到A B其中一個list結束後把另一個剩餘的複製進res list中

以下為簡單圖示說明



```

1 Node* plus(Node *A_head , Node* B_head){
2     //創一個reslist來接
3     Node *res = NULL, **curr = &res;
4     while(A_head && B_head){
5         if(A_head->expo == B_head->expo){
6             //如果相加係數不等於0創新的node
7             if((A_head->coef + B_head->coef) != 0){
8                 *curr = new_node((A_head->coef + B_head->coef), A_head->expo);
9                 curr = &(*curr)->next;
10            }
11            A_head = A_head->next;
12            B_head = B_head->next;
13        }
14        else if(A_head->expo > B_head->expo){
15            *curr = new_node(A_head->coef, A_head->expo);
16            curr = &(*curr)->next;
17            A_head = A_head->next;
18        }
19        else {
20            *curr = new_node(B_head->coef, B_head->expo);
21            curr = &(*curr)->next;
22            B_head = B_head->next;
23        }
24    }
25    //A_head 或 B_head有一個不存在時 代表一個讀完了 把另一個的後面全部複製進去res就好
26    while(!A_head && B_head){
27        *curr = new_node(B_head->coef, B_head->expo);
28        curr = &(*curr)->next;
29        B_head = B_head->next;
30    }
31    while(!B_head && A_head){
32        *curr = new_node(A_head->coef, A_head->expo);
33        curr = &(*curr)->next;
34        A_head = A_head->next;
35    }
36    return res;
37 }
38

```

Node* minus(Node * A_head , Node* B_head)

大致的邏輯概念和plus差不多 僅更動B的coef正負號

```
1 Node* minus(Node *A_head , Node* B_head){
2     Node *res = NULL, **curr = &res;
3     while(A_head && B_head){
4         if(A_head->expo == B_head->expo){
5             //如果相加係數不等於0創新的node
6             if((A_head->coef - B_head->coef) != 0){
7                 *curr = new_node((A_head->coef - B_head->coef), A_head->expo);
8                 curr = &(*curr)->next;
9             }
10            A_head = A_head->next;
11            B_head = B_head->next;
12        }
13        else if(A_head->expo > B_head->expo){
14            *curr = new_node(A_head->coef, A_head->expo);
15            curr = &(*curr)->next;
16            A_head = A_head->next;
17        }
18        else {
19            *curr = new_node(-(B_head->coef), B_head->expo);
20            curr = &(*curr)->next;
21            B_head = B_head->next;
22        }
23    }
24    //A_head 或 B_head有一個不存在時 代表一個讀完了 把令一個的後面全部複製進去res就好
25    while(B_head){
26        *curr = new_node(-(B_head->coef), B_head->expo);
27        curr = &(*curr)->next;
28        B_head = B_head->next;
29    }
30    while(A_head){
31        *curr = new_node(A_head->coef, A_head->expo);
32        curr = &(*curr)->next;
33        A_head = A_head->next;
34    }
35    return res;
36 }
```

結果截圖

以下維test1~test5的輸出結果

```
test1.txt

A:
(3)X^90 + (2)X^5 + (10)X^0
B:
(5)X^40 + (3)X^20 + (6)X^3 + (10)X^0
A+B:
(3)X^90 + (5)X^40 + (3)X^20 + (2)X^5 + (6)X^3 + (20)X^0
A-B:
(3)X^90 + (-5)X^40 + (-3)X^20 + (2)X^5 + (-6)X^3

test2.txt

A:
(10)X^100 + (5)X^50 + (3)X^1 + (99)X^0
B:
(10)X^100 + (5)X^50 + (3)X^1 + (99)X^0
A+B:
(20)X^100 + (10)X^50 + (6)X^1 + (198)X^0
A-B:
0

test3.txt

A:
(9)X^10 + (89)X^4 + (85)X^2 + (10)X^0
B:
(10)X^11 + (10)X^6 + (89)X^4 + (-5)X^2
A+B:
(10)X^11 + (9)X^10 + (10)X^6 + (178)X^4 + (80)X^2 + (10)X^0
A-B:
(-10)X^11 + (9)X^10 + (-10)X^6 + (90)X^2 + (10)X^0

test4.txt

A:
(99)X^10 + (98)X^5 + (97)X^1 + (96)X^0
B:
(98)X^10 + (97)X^5 + (96)X^1 + (95)X^0
A+B:
(197)X^10 + (195)X^5 + (193)X^1 + (191)X^0
A-B:
(1)X^10 + (1)X^5 + (1)X^1 + (1)X^0

test5.txt

A:
(-1)X^50 + (-3)X^25 + (-6)X^0
B:
(-10)X^51 + (-3)X^25 + (-14)X^0
A+B:
(-10)X^51 + (-1)X^50 + (-6)X^25 + (-20)X^0
A-B:
(10)X^51 + (-1)X^50 + (8)X^0
```