資料結構　Homework#9 F74109016 葉惟欣

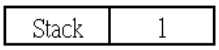
```c
//F74109016_葉惟欣 Finished in 2021/10/18 Homework#9
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef int bool;
enum { false, true };
struct node
{
   int data;
   struct node *right_link;
   struct node *left_link;
   struct node *back;
};
struct link_list{
    int data;
    struct link_list *back;
    struct link_list *front;
};
typedef struct node NODE;
typedef struct link_list LINK_LIST;
NODE* read(int arr[50],int len){
    int i=0;
    bool child = false;
    NODE *first,*new_NODE;
    NODE *current =(NODE *)malloc(sizeof(NODE));

    while(i<len){
        new_NODE= (NODE *)malloc(sizeof(NODE));
        new_NODE->left_link = NULL;
        new_NODE->right_link = NULL;
        new_NODE->back = NULL;
        if(i==0){
            new_NODE->data = arr[i];
            current = new_NODE;
            first = current;
            i++;
        }
        else{
            if(current->left_link == NULL){  //左節點還沒滿
                new_NODE->data = arr[i];
                current->left_link = new_NODE;
                new_NODE->back = current;
                current = new_NODE;
                if(new_NODE->data == -1){
                    current = current->back;
                }
                i++;
            }
            else if(current->right_link == NULL){ //右節點還沒滿
                new_NODE->data = arr[i];
                current->right_link = new_NODE;
                new_NODE->back = current;
                current = new_NODE;
                if(new_NODE->data == -1){
                    current = current->back;
                }
                i++;
            }
            else{
                current = current->back;
            }
        }
    }
    return first;
}
```

描述非遞迴的後序採訪實作方法: 程式碼:66~95

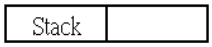
```
66  void Nonpostorder(NODE * first,int length){
67      NODE *S[length];
68      LINK_LIST * result=(LINK_LIST *)malloc(sizeof(LINK_LIST));
69      result->front=NULL;
70      result->data =0;
71      int top = -1;
72      NODE *current = first; //current是root
73      S[++top] = first;       //stack 裡面放root
74      while(top!=-1){
75          current = S[top--]; //將stack 現在放出來
76          LINK_LIST *new_result =(LINK_LIST *)malloc(sizeof(LINK_LIST)); //產生一個result
77          new_result->data = current->data; //將放出來的放到result
78          if(result!=NULL){
79          result->back = new_result;
80          new_result->front = result; //互指
81          }
82          result = new_result;
83          if(current->left_link->data!=-1){
84              S[++top] = current->left_link;
85          }
86          if(current->right_link->data!=-1){
87              S[++top] = current->right_link;
88          }
89      }
90      int i;
91      do{
92          printf("%d ",result->data);
93          result = result->front;
94      }while(result->front!=NULL);
95  }
```

一開始先將 NODE* current 指向 NODE* first

將 NODE * first 放入 stack

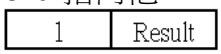


進入迴圈



放出1 current指向1

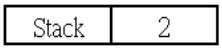
放出 stack 的最後一個節點，並將將 NODE * current 指向他



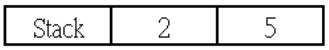
將1放入Result

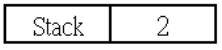
將 NODE * current 的值放入 result 的鏈結串列。

如果 current 有左節點的話放入 stack。



如果 current 有右節點的話放入 stack。



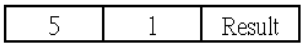


放出5 current指向5

再一次迴圈

放出 stack 的最後一個節點，並將將 NODE * current 指向他

將 NODE * current 的值放入 result 的鏈結串列。

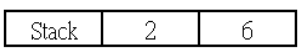


將5放入Result

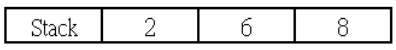
將 result 與新節點互指形成 duble linked list。

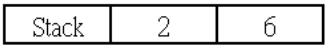
將 result 指向新的 result 節點

如果 current 有左節點的話放入 stack。



如果 current 有右節點的話放入 stack。



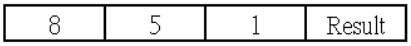


放出8 current指向8

再一次迴圈

放出 stack 的最後一個節點，並將將 NODE * current 指向他

將 result 與新節點互指形成 duble linked list。



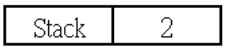
將8放入Result

將 NODE * current 的值放入 result 的鏈結串列。

將 result 指向新的 result 節點

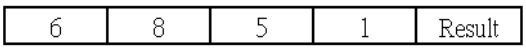
8 無任何 child

再一次迴圈



放出6 current指向6

放出 stack 的最後一個節點，並將將 NODE * current 指向他



將6放入Result

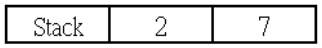
將 NODE * current 的值放入 result 的鏈結串列。

將 result 與新節點互指形成 duble linked list。

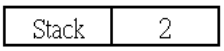
將 result 指向新的 result 節點

如果 current 有左節點的話放入 stack。

如果 current 有右節點的話放入 stack。

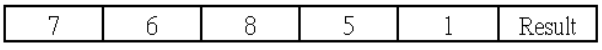


再一次迴圈



放出7 current指向7

放出 stack 的最後一個節點，並將將 NODE * current 指向他



將7放入Result

將 NODE * current 的值放入 result 的鏈結串列。

將 result 與新節點互指形成 duble linked list。

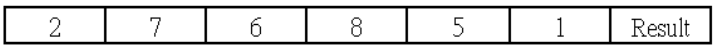
將 result 指向新的 result 節點

7 無任何 child

再一次迴圈

放出2 current指向2

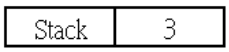
放出 stack 的最後一個節點，並將將 NODE * current 指向他



將2放入Result

將 NODE * current 的值放入 result 的鏈結串列。

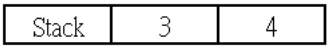
將 result 與新節點互指形成 duble linked list。



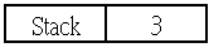
將 result 指向新的 result 節點

如果 current 有左節點的話放入 stack。

如果 current 有右節點的話放入 stack。



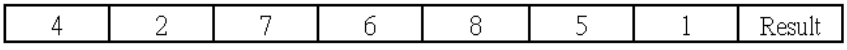
再一次迴圈



放出4 current指向4

放出 stack 的最後一個節點，並將將 NODE * current 指向他

將 NODE * current 的值放入 result 的鏈結串列。



將4放入Result

將 result 與新節點互指形成 duble linked list。

將 result 指向新的 result 節點

4 無任何 child

再一次迴圈　　Stack　　放出3 current指向3
放出 stack 的最後一個節點，並將將 NODE * current 指向他
將 NODE * current 的值放入 result 的鏈結串列。

3	4	2	7	6	8	5	1	Result

將3放入Result

將 result 與新節點互指形成 duble linked list。
將 result 指向新的 result 節點
最後再從現在 NODE *result 印回去
印出 34276851

描述非遞迴的前序採訪實作方法: 程式碼:96~113

```
96  void Nonpreorder(NODE * first,int length){
97      NODE *Stack[length];
98      NODE *current;
99      int top = -1;
100     Stack[++top]= first;
101     while(top!=-1){
102         current = Stack[top--];
103         if(current->right_link->data!=-1){
104             Stack[++top] = current->right_link;
105         }
106         if(current->left_link->data!=-1){
107             Stack[++top] = current->left_link;
108         }
109         if(current->data!=-1){
110             printf("%d ",current->data);
111         }
112     }
113 }
```

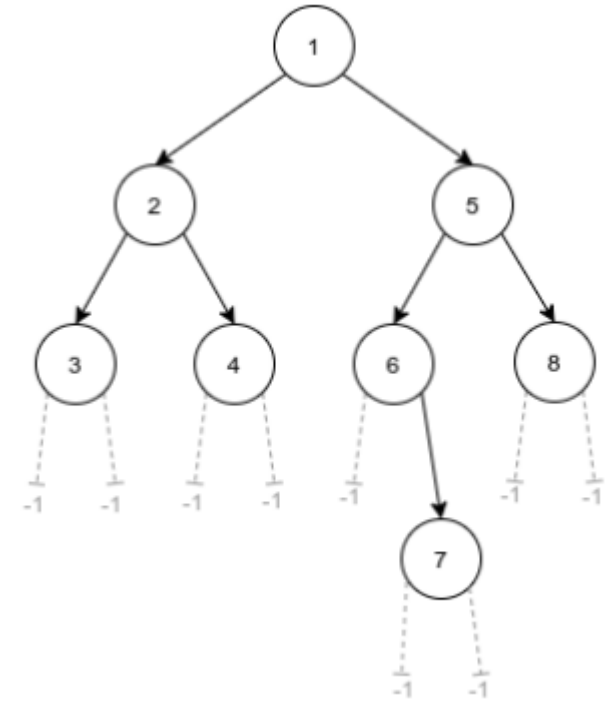
剛開始有一個陣列為 stack 裡面存放的元素都是 NODE，長度為輸入字串中非-1 的整數。起初就先將第一個 NODE 元素放入 stack。
開始跑迴圈。當 stack 不為空時(top=-1)，迴圈不會結束。
迴圈內部:
　　將 stack 陣列最後一個元素放出➔pop。將 current 指向那個元素。
如果 current 指向的那個元素有右小孩的話，將右小孩放入 stack，
如果 current 指向的那個元素有左小孩的話，將左小孩放入 stack。
如果都沒有小孩的話就印出來。

當 stack 不為空時，就再印出 stack 陣列的最後一個元素。將 current 指向那個元素。

以提供的公開測資作舉例。

```
17
1 2 3 -1 -1 4 -1 -1 5 6 -1 7 -1 -1 8 -1 -1
```

將輸入的字串建立成 tree，進行非遞迴 preorder 的走訪。



剛開始有一個陣列為 stack 裡面存放的元素都是 NODE，長度為輸入字串中非-1 的整數。起初就先將第一個 NODE 元素放入 stack。

Stack	1

開始跑迴圈。當 stack 不為空時(top=-1)，迴圈不會結束。
迴圈內部:

Stack	

放出1 current指向1

將 stack 陣列最後一個元素放出➔pop。將 current 指向那個元素。
如果 current 指向的那個元素有右小孩的話，將右小孩放入 stack，

Stack	5

如果 current 指向的那個元素有左小孩的話，將左小孩放入 stack。

Stack	5	2

如果都沒有小孩的話就印出 current 所指向的 1 來。

Stack	5

放出2 current指向2

將 stack 陣列最後一個元素放出➔pop。將 current 指向那個元素。
如果 current 指向的那個元素有右小孩的話，將右小孩放入 stack，

Stack	5	4

如果 current 指向的那個元素有左小孩的話，將左小孩放入 stack。
如果都沒有小孩的話就印出 current 所指向的 2 來。

Stack	5	4	3

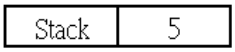
將 stack 陣列最後一個元素放出➔pop。將 current 指向那個元素。

Stack	5	4

放出3 current指向3

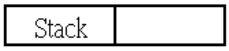
如果 current 指向的那個元素有右小孩的話，將右小孩放入 stack，
如果 current 指向的那個元素有左小孩的話，將左小孩放入 stack。

如果都沒有小孩的話就印出 current 所指向的 3 來。



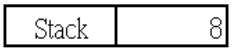
放出4 current指向4

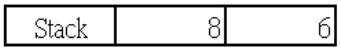
將 stack 陣列最後一個元素放出➔pop。將 current 指向那個元素。
如果 current 指向的那個元素有右小孩的話，將右小孩放入 stack，
如果 current 指向的那個元素有左小孩的話，將左小孩放入 stack。
如果都沒有小孩的話就印出 current 所指向的 4 來。

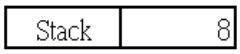


放出5 current指向5

將 stack 陣列最後一個元素放出➔pop。將 current 指向那個元素。
如果 current 指向的那個元素有右小孩的話，將右小孩放入 stack，
如果 current 指向的那個元素有左小孩的話，將左小孩放入 stack。
如果都沒有小孩的話就印出 current 所指向的 5 來。

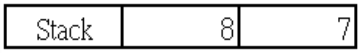


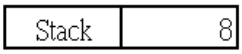




放出6 current指向6

將 stack 陣列最後一個元素放出➔pop。將 current 指向那個元素。
如果 current 指向的那個元素有右小孩的話，將右小孩放入 stack，
如果 current 指向的那個元素有左小孩的話，將左小孩放入 stack。
如果都沒有小孩的話就印出 current 所指向的 6 來。





放出7 current指向7

將 stack 陣列最後一個元素放出➔pop。將 current 指向那個元素。
如果都沒有小孩的話就印出 current 所指向的 7 來。

放出8 current指向8

將 stack 陣列最後一個元素放出➔pop。將 current 指向那個元素。
如果都沒有小孩的話就印出 current 所指向的 8 來。

```
114 int main(){
115     NODE *first =(NODE *)malloc(sizeof(NODE));
116     NODE *second =(NODE *)malloc(sizeof(NODE));
117     char string [50];
118     int len,i;
119     scanf("%d",&len);
120     int string1 [len];
121     int length=1;
122     for(i=0;i<len;i++){
123         scanf("%s",string);
124         string1[i] = atoi(string);
125         if(string1[i]!=-1){
126             length++;
127         }
128     }
129     first = read(string1,len);
130     Nonpreorder(first,len);
131     printf("\n");
132     Nonpostorder(first,len);
133
134 }
```