\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Report: HW8

Author: F74071027 學貝萁 <youqing1211@gmail.com>

Class: 甲班

Description:

　　指標的進階用法──Linked list，也是一種資料結構，改善了陣列大小不能更改的缺點，在改變順序上也有更好的做法。透過這次作業為Linked list寫一個quick sort，可以直接了解Linked list的概念與學會如何建造Linked list，還有如何在Linked list中新建、串接或刪除一個或多個Node。如果懂quick sort的概念就當複習，不懂就當學習嚕。

　　由於Linked list本身的結構關係，malloc和structure都扮演著很重要的角色，透過malloc才能取得一塊存資料的記憶體，透過structure才可以在一個node中存放資料與指向下一個node的位址，需特別注意。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Code:

#include<stdio.h>

#include<stdlib.h>

int i;

typedef struct node{

int value;

struct node \*next;

} Node;

Node \*SLL\_build();

Node \*SLL\_qsort();

void \*SLL\_qsort2();

void SLL\_print();

void SLL\_delete();

int main(int argc, char \*argv[]){

srand(atoi(argv[3]));

Node \*llist = SLL\_build(atoi(argv[1]), atoi(argv[2]));

printf("Original SLL:");

SLL\_print(llist);

llist = SLL\_qsort(llist);

printf("Sorted SLL:");

SLL\_print(llist);

SLL\_delete(llist);

return 0;

}

Node \*SLL\_build(int n, int m){

Node \*node\_head = malloc(sizeof(Node));

Node \*node\_tag = node\_head, \*node\_curr;

for(i=0;i<n;i++){

node\_curr = malloc(sizeof(Node));

node\_tag->next = node\_curr;

node\_curr->next = NULL;

node\_tag = node\_curr;

}

Node \*tmp = node\_head;

tmp->value = -1;

tmp = tmp->next;

while(tmp!=NULL) tmp->value = rand() % m, tmp = tmp->next;

return node\_head;

}

Node \*SLL\_qsort(Node \*node\_head){

Node \*node\_last = node\_head;

while(node\_last->next!=NULL)node\_last = node\_last->next;

SLL\_qsort2(node\_head, node\_last);

return node\_head;

}

void \*SLL\_qsort2(Node \*head, Node \*tag){

if(head->next==tag){

if((head->value)>(tag->value)){

int a = head->value;

head->value = tag->value;

tag->value = a;

}

}

else if(head==tag){}

else{

Node \*last = tag, \*cur = head, \*tmp;

while(cur->next!=tag){

if((cur->next->value)>(tag->value)){

tmp = cur->next;

cur->next = cur->next->next;

tmp->next = last->next;

last->next = tmp;

last = tmp;

}

else cur = cur->next;

}

SLL\_qsort2(head, cur);

SLL\_qsort2(tag, last);

}

}

void SLL\_print(Node \*head){

Node \*tmp = head->next;

while(tmp!=NULL){

printf(" %d", tmp->value);

tmp = tmp->next;

}

printf("\n");

}

void SLL\_delete(Node \*head){

Node \*cur = head, \*tmp;

while(cur!=NULL){

tmp = cur->next;

free(cur);

cur = tmp;

}

}

Compilation:

gcc hw8.c -o hw8

Execution:

./hw8 10 10 400

Output:

Original SLL: 5 1 2 2 0 2 8 7 5 9

Sorted SLL: 0 1 2 2 2 5 5 7 8 9