Data Structure Homework 1

繳交期限： 10/8 17:00 前
補交期限(7 折)： 10/15 17:00 前

手寫題：

13. If the efficiency of the algorithm `doIt` can be expressed as $O(n) = n^2$, calculate the efficiency of the following program segment:

```
for (i = 1; i < n; i *= 2)
    doIt (...)
```

17. An algorithm processes a given input of size n. If n is 4096, the run time is 512 milliseconds. If n is 16,384, the run time is 2048 milliseconds. What is the efficiency? What is the big-O notation?

20. Three students wrote algorithms for the same problem. They tested the three algorithms with two sets of data as shown below:

    a. Case 1: $n = 10$
    - Run time for student 1: 1
    - Run time for student 2: 1/100
    - Run time for student 3: 1/1000

    b. Case 2: $n = 100$
    - Run time for student 1: 10
    - Run time for student 2: 1
    - Run time for student 3: 1

    What is the efficiency for each algorithm? Which is the best? Which is the worst? What is the minimum number of test cases (n) in which the best algorithm has the best run time?

程式題：

32. Rewrite Program 1-4 to create a list of nodes. Each node consists of two fields. The first field is a pointer to a structure that contains a student id (integer) and a grade-point average (float). The second field is a link. The data are to be read from a text file.

    Then write a program to read a file of at least 10 students and test the function you wrote. You will also need to use the generic compare code in Program 1-6 in your program.

PROGRAM 1-4 Create List with Two Linked Nodes

```
 1  /* Create a list with two linked nodes.
 2        Written by:
 3        Date:
 4  */
 5  #include <stdio.h>
 6  #include <stdlib.h>
 7  #include "P1-02.h"                        // Header file
 8
 9  int main (void)
10  {
11  // Local Definitions
12     int*  newData;
13     int*  nodeData;
14     NODE* node;
15
16  // Statements
17     newData  = (int*)malloc (sizeof (int));
18     *newData = 7;
19     node = createNode (newData);
20
21     newData    = (int*)malloc (sizeof (int));
22     *newData   = 75;
23     node->link = createNode (newData);
24
25     nodeData = (int*)node->dataPtr;
26     printf ("Data from node 1: %d\n", *nodeData);
27
28     nodeData = (int*)node->link->dataPtr;
29     printf ("Data from node 2: %d\n", *nodeData);
30     return 0;
31  }  // main
```

```
Results:
Data from node 1: 7
Data from node 2: 75
```

PROGRAM 1-6 Compare Two Integers

```
/* Demonstrate generic compare functions and pointer to
   function.
      Written by:
      Date:
*/
#include <stdio.h>
#include <stdlib.h>
#include "P1-05.h"                          // Header file

int   compare (void* ptr1, void* ptr2);

int main (void)
{
// Local Definitions

   int i = 7 ;
   int j = 8 ;
   int lrg;

// Statements
   lrg = (*(int*) larger (&i, &j, compare));

   printf ("Larger value is: %d\n", lrg);
   return 0;
}  // main
/* ==================== compare ====================
   Integer specific compare function.
      Pre  ptr1 and ptr2 are pointers to integer values
      Post returns +1 if ptr1 >= ptr2
           returns -1 if ptr1 <  ptr2
*/
int compare (void* ptr1, void* ptr2)
```

continued

PROGRAM 1-6 Compare Two Integers *(continued)*

```
{
  if (*(int*)ptr1 >=  *(int*)ptr2)
     return 1;
  else
     return -1;
}  // compare
```

```
Results:
Larger value is: 8
```