2022 Fall Data Structure HW1

Handwriting

1. (25%) Determine the big-O notation for the following:

   a. $5n^{5/2} + n^{2/5}$
   b. $6\log(n) + 9n$
   c. $3n^4 + n\log(n)$
   d. $5n^2 + n^{3/2}$

2. (25%) 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++)
    for (j = 1; j < n, j++)
         doIt (...)
```

3. (25%) Given that the efficiency of an algorithm is $5n^2$, if a step in this algorithm takes 1 nanosecond (10^{-9} seconds), how long does it take the algorithm to process an input of size 1000?

4. (25%) Write a compare function (see Program 1-6) to compare two strings.

Programming

5. (100%) 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 (integer). 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, at most 100 students, and test the function you wrote. Print the max score of these student and his/her id.
   (If two or above students have the same grades, then print the smallest id)
   Below is the example (input.txt is in E3):

Input Format:	Output Format:

311511037 88
325621523 66
333625847 12
325896315 60
333156945 90
311510088 76
308616235 63
315884463 28
333333333 33
111111111 100
	Maximum ID: 111111111, Maximum score: 100

Appendix.

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

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

larger function in Program 1-6:

```
void* larger (void* dataPtr1,    void* dataPtr2,
               int (*ptrToCmpFun)(void*, void*))
{
   if ((*ptrToCmpFun) (dataPtr1, dataPtr2) > 0)
         return dataPtr1;
   else
         return dataPtr2;
}  // larger
```

PROGRAM 1-4 Create List with Two Linked Nodes

```
/* Create a list with two linked nodes.
      Written by:
      Date:
*/
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"                          // Header file

int main (void)
{
// Local Definitions
   int*  newData;
   int*  nodeData;
   NODE* node;

// Statements
   newData  = (int*)malloc (sizeof (int));
   *newData = 7;
   node = createNode (newData);

   newData    = (int*)malloc (sizeof (int));
   *newData   = 75;
   node->link = createNode (newData);

   nodeData = (int*)node->dataPtr;
   printf ("Data from node 1: %d\n", *nodeData);

   nodeData = (int*)node->link->dataPtr;
   printf ("Data from node 2: %d\n", *nodeData);
   return 0;
}  // main
```

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

Node definition & creation code:

```
typedef struct node
{
          void* dataPtr;
  struct node* link;
} NODE;

/* =================== createNode ====================
   Creates a node in dynamic memory and stores data
   pointer in it.
      Pre  itemPtr is pointer to data to be stored.
      Post node created and its address returned.
*/
NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr = (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link    = NULL;
   return nodePtr;
}  // createNode
```