CSD101: Introduction to Computing and Programming

Lab Quiz: Quiz #7

```
Max marks:  40+10 bonus marks                                              17-11-2020
Time:  10am-11am
```

1. *You should upload your code and sample input.txt file in a zip archive on Blackboard.*
2. *You get 10 bonus marks if you complete both the iterative and recursive versions. The normalized score will be calculated by* ($\frac{\textit{Marks obtained}}{40} \times 10$). *So, if you do any one out of iterative or recursive correctly you get 40 marks (the max marks). You get bonus marks only if you do both iterative and recursive correctly.*

1. This lab quiz is slightly different from the earlier one. You have been given a code template and you have to fill in code for the missing parts that are indicated via comments in the code so that the entire program works correctly.

   The C program given implements merging of two sorted lists of integers that have been sorted in ascending order. For example, if the two lists are:

   `[ 1 4 8 9 14 ]` and `[ 2 6 9 12 ]` then the merged list is `[ 1 2 4 6 8 9 9 12 14 ]`.

   To create the initial lists the program reads the integer values from the file `input.txt`. Note that the input must have values already sorted in ascending order. In the `input.txt` file the values for each list are available separated by a space and ending with a newline.

   The output is printed on the terminal in three/four lines: i) first line: first input list ii) second line: second input list iii) third/fourth line: merged list(s). The merge should not change the two input lists. Ensure that you take care of special or corner cases.

   Some sample inputs and outputs are given below. The input is actually in the file `input.txt`.

   The first two lines are the input lines that are in the file `input.txt`; the next four lines are the output where the first two lines give the two input lists and the third and fourth lines give the merged lists by the two methods.

   If the list is empty then the input line will be a blank line and an empty list should be output as `[ ]`.

   If you do just one of recursive or iterative merge (but not both) you will have only three lines of output.

   Sample 1:

   ```
   1 4 5 7 9 80
   2 3 21 33 45 58
   [ 1 4 5 7 9 80 ]
   [ 2 3 21 33 45 58 ]
   Recursive merge = [ 1 2 3 4 5 7 9 21 33 45 ]
   Iterative merge = [ 1 2 3 4 5 7 9 21 33 45 ]
   ```

Sample 2:

```
-2 5 9 22 34
-8 -6 4 27 28 30 45
[ -2 5 9 22 34 ]
[ -8 -6 4 27 28 30 45 ]
Recursive merge = [ -8 -6 -2 4 5 9 22 27 28 30 34 45 ]
Iterative merge = [ -8 -6 -2 4 5 9 22 27 28 30 34 45 ]
```

The code is given below. It includes some descriptive comments and comments that asks you to fill in code - shown by /* FILL IN CODE HERE */. You have to fill in the code at the places indicated so that the whole program works correctly. If you only want to do one of iterative or recursive comment out or delete the parts that are not relevant.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define N 101 //max length of line in input.txt is N-1

struct Node {
  int data;
  struct Node *next;
};

int isEmpty(struct Node *lp) {return (lp==NULL);}

void printList(struct Node *lp) {
  if (isEmpty(lp)) printf("[ ]\n");
  else {
    struct Node *tmp=lp;
    printf("[ ");
    while (tmp!=NULL) {printf("%d ", tmp->data); tmp=tmp->next;}
    printf("]\n");
  }
  return;
}

/* We create a QNode structure using Node that has a pointer to
the end of the list like a queue along with some essential
functions that manipulate it so that it is easier to write merge. */

struct QNode {
  int data;
  struct Node *head, *tail;
};
```

```
struct QNode *createQN(void) {
  //creates an empty QNode
  struct QNode *qp=malloc(sizeof(struct QNode));
  qp->head=qp->tail=NULL;
  return qp;
}

int isEmptyQN(struct QNode *qp) {return (qp->head==NULL && qp->tail==NULL);}

struct QNode *addLast(struct QNode *qp, int info) {
  //adds a new Node at the end of qp with data field equal to info
  struct Node *newp=malloc(sizeof(struct Node));
  newp->data=info;
  newp->next=NULL;
  if (isEmptyQN(qp)) {
    qp->head=qp->tail=newp;
  }
  else {
    qp->tail->next=newp;
    qp->tail=newp;
  }
  return qp;
}
/* mergefn is the recursive helper function for a recursive merge */
struct QNode *mergefn(struct Node *lp1, struct Node *lp2, struct QNode *qp) {

  /* FILL IN CODE HERE */

}

struct QNode *mergeRecursive(struct Node *lp1, struct Node *lp2) {
  //Recursive version of merge using mergefn
  return mergefn(lp1, lp2, createQN());
}

/* This merge is an iterative merge. */
struct QNode *mergeIterative(struct Node *lp1, struct Node *lp2) {

 /* FILL IN CODE HERE */
}
```

```
int main(void) {
  FILE *infile=fopen("input.txt", "r");
  struct QNode *qp1=createQN(), *qp2=createQN();
  char str[N], s[10];
  int ind;
  // Creates first list
  fgets(str, N, infile);//reads line
  ind=0;
  while (isspace(str[ind])) ind++;//purge initial white space
  while (str[ind]!='\0') {//enters with non-whitespace char
    int i=0;
    while (!isspace(str[ind+i])) {s[i]=str[ind+i]; i++;}//read digits
    s[i]='\0';
    qp1=addLast(qp1, atoi(s));//add integer to list
    ind+=i;
    while (isspace(str[ind])) ind++;//purge whitespace
  }
  // Creates second list

  /* FILL IN CODE HERE FOR CREATING THE SECOND LIST */

  fclose(infile);
  //print list 1
  printList(qp1->head);
  //print list 2
  printList(qp2->head);
  printf("Recursive merge = ");
  printList(/* FILL IN CODE HERE */);
  printf("Iterative merge = ");
  printList(/* FILL IN CODE HERE */);
  return 0;
}
```

[40+10 Bonus]