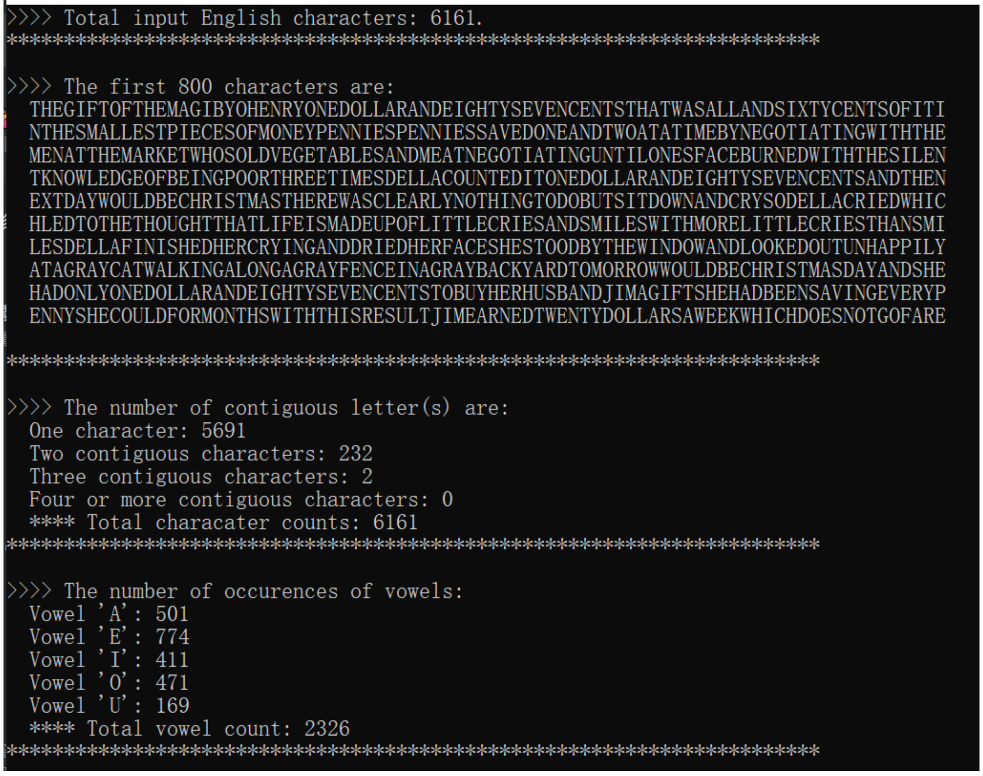
Spring 2023, ISTM, Purdue-FCU 2+2 ECE Program
Advanced C Programming, Quiz 1

Total FIVE FILES for Quiz 1. Use file name quiz1_dxxxxxxx_1.c for Question 1 and file names quiz1_dxxxxxxx_2.dev, quiz1_dxxxxxxx_2.h, quiz1_dxxxxxxx_2.c, and quiz1_dxxxxxxx_2_main.c for Question 2, where dxxxxxxx is your student ID. When you finish a question, submit all the above files to the instructor's computer.

1. (40 points) Start with program skeleton **quiz1_skeleton_1.c** and change the file name to **quiz1_dxxxxxxx_1.c**. Write a C program to perform the following steps:
   1) Let capacity be the size of a data buffer with initial value 512. Declare a pointer to character "**char** *buffer;" and dynamically allocate 512 characters as the initial memory space of buffer[].
   2) Open the input text file file Gift_of_Magi.txt and read the file using fgetc() to input the file character by character until reaching the end of file.
      a. If the input character is an English letter, convert it to uppercase letter and store in buffer[], starting from index 0. When the memory space of buffer[] becomes full, extend the size of buffer 512 bytes more.
      b. If the input character is not an English letter, ignore it.
   3) Insert end-of-string '\0' following the last input character. Print the number of input English letters, i.e., the length of string.
   4) Open an output text file "result.txt" and write the file using fwrite().
   5) Print the first 800 characters of the input text on the console, 80 characters in a line.
   6) Count and report the number of one character letter, two contiguous character letters, three contiguous character letters, and four or more contiguous character letters.
   7) Count and report the number of each occurrence of vowels, 'A', 'E', 'I', 'O', and 'U'. Print the total vowel count.
   8) Release memory space of buffer[].

In steps 7 and 8, use string functions in library <string.h> and **NOT** to use library <ctype.h>. Program execution example:



```
>>>> Total input English characters: 6161.
************************************************************************

>>>> The first 800 characters are:
  THEGIFTOFTHEMAGIBYOHENRYONEDOLLARANDEIGHTYSEVENCENTSTHATWASALLANDSIXTYCENTSOFITI
  NTHESMALLESTPIECESOFMONEYPENNIESPENNIESSAVEDONEANDTWOATATIMEBYNEGOTIATINGWITHTHE
  MENATTHEMARKETWHOSOLDVEGETABLESANDMEATNEGOTIATINGUNTILONESFACEBURNEDWITHTHESILEN
  TKNOWLEDGEOFBEINGPOORTHREETIMESDELLACOUNTEDITONEDOLLARANDEIGHTYSEVENCENTSANDTHEN
  EXTDAYWOULDBECHRISTMASTHEREWASCLEARLYNOTHINGTODOBUTSITDOWNANDCRYSODELLACRIEDWHIC
  HLEDTOTHETHOUGHTTHATLIFEISMADEUPOFLITTLECRIESANDSMILESWITHMORELITTLECRIESTHANSMI
  LESDELLAFINISHEDHERCRYINGANDDRIEDHERFACESHESTOODBYTHEWINDOWANDLOOKEDOUTUNHAPPILY
  ATAGRAYCATWALKINGALONGAGRAYFENCEINAGRAYBACKYARDTOMORROWWOULDBECHRISTMASDAYANDSHE
  HADONLYONEDOLLARANDEIGHTYSEVENCENTSTOBUYHERHUSBANDJIMAGIFTSHEHADBEENSAVINGEVERYP
  ENNYSHECOULDFORMONTHSWITHTHISRESULTJIMEARNEDTWENTYDOLLARSAWEEKWHICHDOESNOTGOFARE

************************************************************************

>>>> The number of contiguous letter(s) are:
  One character: 5691
  Two contiguous characters: 232
  Three contiguous characters: 2
  Four or more contiguous characters: 0
  **** Total characater counts: 6161
************************************************************************

>>>> The number of occurences of vowels:
  Vowel 'A': 501
  Vowel 'E': 774
  Vowel 'I': 411
  Vowel 'O': 471
  Vowel 'U': 169
  **** Total vowel count: 2326
************************************************************************
```

(to be continued)

2. (60 points) Create a new Dev-C++ console project **quiz1_dxxxxxxx_2.dev** and add the three files **quiz1_skeleton_2.h**, **quiz1_skeleton_2.c**, and **quiz1_skeleton_2_main.c** to the project. Rename these three files to **quiz1_dxxxxxxx_2.h**, **quiz1_dxxxxxxx_2.c**, and **quiz1_dxxxxxxx_2_main.c**. Remember to change the file name in directive **#include "quiz1_skeleton_2.h"**. The header file **quiz1_skeleton_2.h** is the specification of a ***non-ordered*** and ***non-duplicate*** linear list using fixed-size array and is defined as the following data structure:

```
typedef int ElemType; // Integer element type.

typedef struct {
   ElemType elem[100]; // fixed-size array of 100 elements.
   int size; // number of elements in the linear list,
} List;
```

The followings are function specification of List operations:

```
// Initialize a linear list, set its size to 0.
void initial(List *);

// The length of a linear list, returns the number of elements, namely size.
int getSize(List);

// Get the element at a position from a linear list, return the designated element.
ElemType getElem(List , int);

// Set the element at a position in a linear list to a specific element.
ElemType setElem(List L, ElemType, int);

// Find the position of an element in L. If successful, return the position of the
// element; otherwise, returns -1.
int search(List, ElemType);

// Insert an element after the end of a linear list, return the position of the
// inserted element. If the inserted element exists in the linear list or the linear list
// is overflow, insertElem() fails, and returns -1.
int insertElem(List *, ElemType);

// Delete an element from a list. If the element is in the linear list, delete it and
// return its position; otherwise, return -1.
int deleteElem(List *, ElemType);

// Print all elements of the linear list starting from the head.
int printList(List);
```

Implement all the above functions in **quiz1_skeleton_2.c**. Assume the linear list is an abstract data type such that its implementation is hidden from the application programmer. That is, the main program cannot access the data structure of linear lists directly. In the main program **quiz1_dxxxxxxx_2_main.c**, write a C program to perform the following steps:

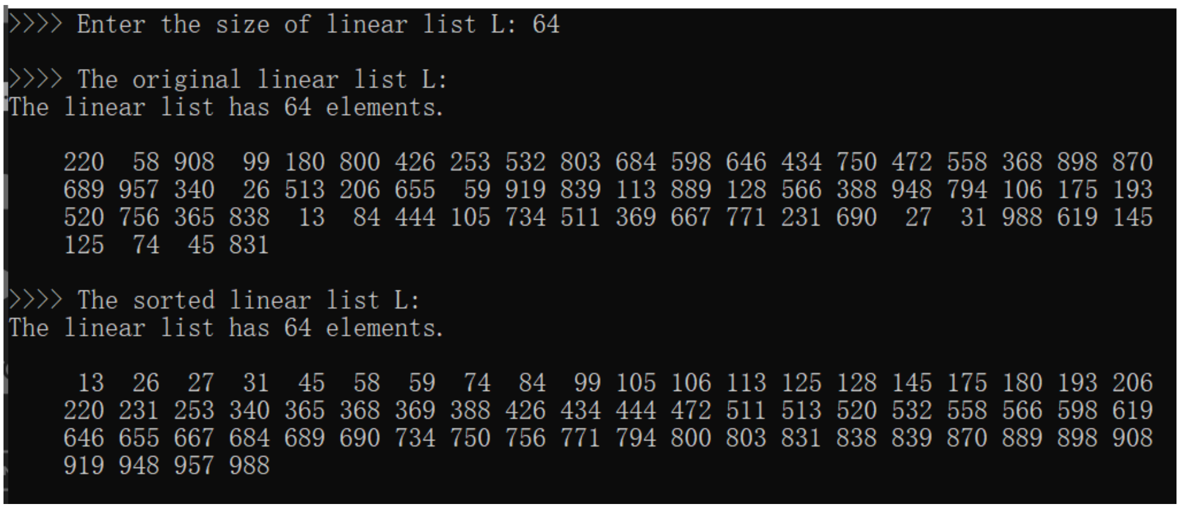
(1) Declare a variable L of a linear list and initialize L.
(2) Enter a positive integer n such that 20≤n≤100, and insert n elements of random number between 0 and 999 to linear list L. No duplicated elements are allowed.
(3) Print the elements of linear list L.
(4) Sort the elements of linear list L. You may *only use the functions in the header file*.

(5) Print the elements of linear list L after sorting.

Given an integer array A of n elements, the pseudo code of insertion sort is shown as the following (array subscript are from 0 to n-1):

```
for i from n-1 to 1
   for j from 1 to i
      Compare and swap elements A[j-1] and A[j];
```

Example of program execution:



```
>>>> Enter the size of linear list L: 64

>>>> The original linear list L:
The linear list has 64 elements.

    220  58 908  99 180 800 426 253 532 803 684 598 646 434 750 472 558 368 898 870
    689 957 340  26 513 206 655  59 919 839 113 889 128 566 388 948 794 106 175 193
    520 756 365 838  13  84 444 105 734 511 369 667 771 231 690  27  31 988 619 145
    125  74  45 831

>>>> The sorted linear list L:
The linear list has 64 elements.

     13  26  27  31  45  58  59  74  84  99 105 106 113 125 128 145 175 180 193 206
    220 231 253 340 365 368 369 388 426 434 444 472 511 513 520 532 558 566 598 619
    646 655 667 684 689 690 734 750 756 771 794 800 803 831 838 839 870 889 898 908
    919 948 957 988
```