Spring 2023, ISTM, FCU-Purdue 2+2 ECE Program
ISTM 2572, Advanced C Programming, Midterm Exam

Use file name mexam_dxxxxxxx_1.c for Question 1, file name mexam _dxxxxxxx_2.c for Question 2, and file name mexam _dxxxxxxx_3.c for Question 3 for your source code, where dxxxxxxx is your student ID. When you finish question, submit the above two files to the instructor's computer.

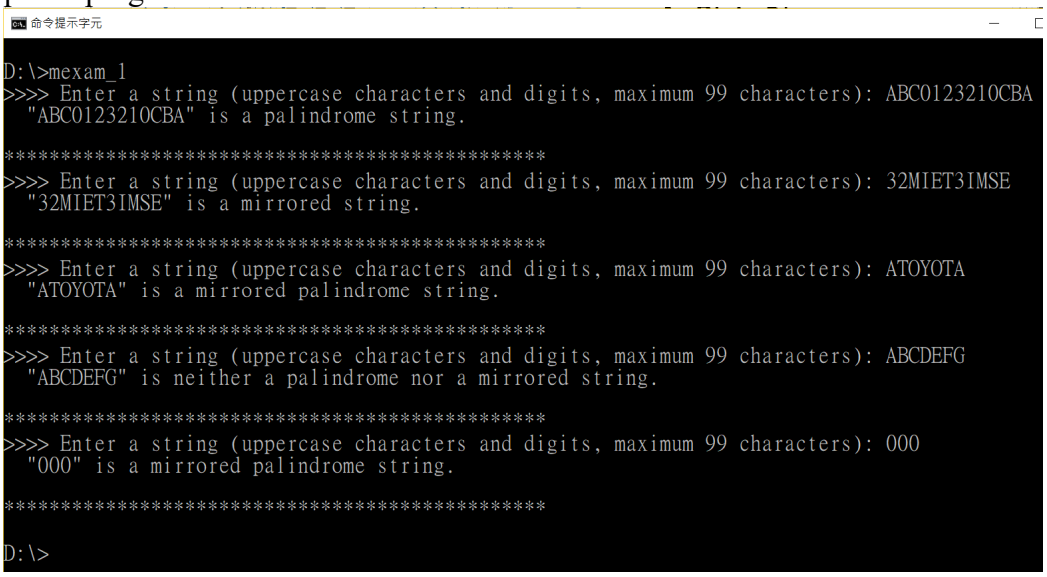
1. (30 points) You may start with program skeleton **mexam_skeleton_1.c** and change the file name to **mexam_dxxxxxxx_1.c**. A ***palindrome*** is a string of (uppercase) letters or digits that is the same as the string reading backward. Next, let us define the ***reverse*** of a letter or a digit, if it exists, as the following table:

Character	Reverse	Character	Reverse	Character	Reverse	Character	Reverse
A	A	K		U	U	4	
B		L	J	V	V	5	Z
C		M	M	W	W	6	
D		N		X	X	7	
E	3	O	O	Y	Y	8	8
F		P		Z	5	9	
G		Q		0	0		
H	H	R		1	1		
I	I	S	2	2	S		
J	L	T	T	3	E		

A ***mirrored*** string is a string of (uppercase) letters and digits if it is same as the string reading backward after each character of the string is replaced by its "reverse", if it exists. For example, "ABC0123210CBA" is a palindrome string; "32MIET3IMSE" is a mirrored string; "ATOYOTA" is a mirrored palindrome string; and "ABCDEFG" is neither a palindrome nor a mirrored string. Write a C program to enter a string of (uppercase) letters or digits and using a **RECURSIVE** function:

```
int check_palindrome_mirrored(char *low, char *high,
                              int is_palindrome, int is_mirrored);
```

to check and report whether the input string is (a) a palindrome, (b) a mirrored string, (c) a mirrored palindrome string, or (d) neither a palindrome nor a mirrored string; repeat the checking operation until the input string is '000". (Hint: File **mexam_skeleton_1.c** contains more detailed explanation of function check_palindrome_mirrored().)
Example of program execution:



(Continue to the next page)

2. (30 points) You may start with program skeleton **mexam_skeleton_2.c** and change the file name to **mexam_dxxxxxxx_2.c**. Define a node type Node and its pointer type NodePtr of a doubly-linked list as below:

```
typedef struct node {
  int data; // data element of a node
  struct node* left; // pointer to the node on the left-hand-side
  struct node* right; // pointer to the node on the right-hand-side
} Node; // type of a doubly-linked node

typedef Node* NodePtr; // type of a node pointer
```

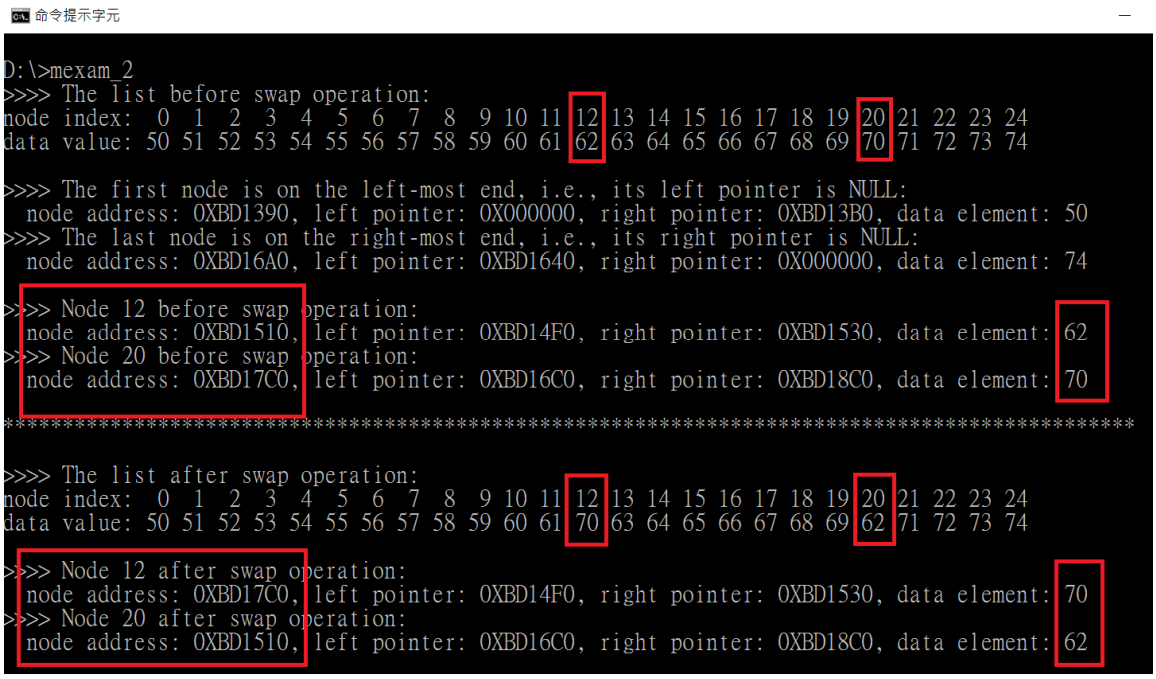
Also doubly-linked list is defined as:

```
typedef struct {
  NodePtr first; // pointer to the first node of the doubly-linked list.
  NodePtr last; // pointer to the last node of the doubly-linked list.
} List; // type of a doubly-linked list
```

Suppose the node inserted at the earlier time is on the left-hand-side of the nodes inserted after. That is, pointer first points to the left-most node and pointer last points to the right-most node. Let L be a doubly-linked list and ptr1 and ptr2 be pointers of two internal nodes, i.e., neither first nor last, in L. Write a C program to implement the following functions:

1. **void** insert(List *L, **int** e): Insert a node of data element e to the last node of list L.
2. **void** swap(NodePtr ptr1, NodePtr ptr2): swap two internal nodes pointed by ptr1 and ptr2. **Do not** simply exchange the data element. The code must "physically" swap the two nodes by *modifying corresponding node pointers*. (Hint: Need to modify eight node pointes.)

Do not change the main program of mexam_skeleton_2.c. Examples of program execution:



(Continue to the next page)

3. (40 points) You may start with program skeleton **mexam_skeleton_3.c** and change the file name to **mexam_dxxxxxxx_3.c**. In image processing, a bitmap is a type of file format to store digital images. The format of a bitmap file is divided into two parts, file header and image information, as defined in the program skeleton. Following the file header and image information, there is the color palette with the size of OffsetBits-InfoSize-14 bytes, and then the image pixel data with the size of ImageSize bytes. The color palette and image pixel data can be declared as pointers of **unsigned char**. The images used in this programming practice does not use color palette, i.e., OffsetBits-InfoSize-14 is 0. The pixel data area is stored using the following format:
   1. Image pixels are stored in the row-major order starting from the bottom-left corner. That is, pixels are stored from left to right and from bottom to top; the pixels in the last row are stored on the front of the pixel data area.
   2. Each pixel takes three bytes representing levels of the three original colors red, green, and blue (RGB) one byte each color. The order of the three bytes is blue color in the first byte, green in the second byte and red in the third byte.
   3. The number of bytes in a row must be a multiple of four. If the number of actual pixel bytes is not a multiple of four, 0X00's are padded at the end of the row.

   Write a C program to perform transformation of mirror reflection along the vertical line on the center of the image and attach the transformed on the right-hand-side of the original image. Output the extended image to a file and print its file header on the screen. Download and use image abraham_lake.bmp to test the program.
   Examples of program execution:

```
命令提示字元

D:\>mexam_3
>>>> File header and image information of the extended image:

Type:           BM
Size:           2160054
Resserved:
OffsetBits:     54
InfoSize:       40
Width:          1600
Height:         450
Planes:         1
BitPerPixel:    24
Compression:    0
ImageSize:      2160000
XResolution:    5669
YResolution:    5669
Colors:         0
ImportantColors:  0
```

   The testing image, abraham_lake.bmp (reduced size):

(Continue to the next page)

The extended image, abraham_lake_vertical_extended.bmp (reduced size):