**STM, FCU-Purdue 2+2 ECE Program**
**ISTM 2731, Advanced C Programming**
**Midterm Exam. Spring 2024**

**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.**

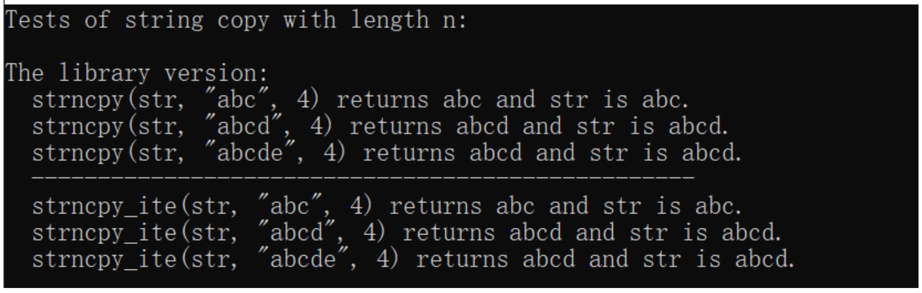
1. (20 points) You may start with program skeleton **mexam_skeleton_1.c** and change the file name to **mexam_DXXXXXXX_1.c**. The following statement is the description of function strncpy() in library <string.h> in C programming language.

   Declaration:
   **char** *strncpy(**char** *str1, **const char** *str2, size_t n);
   Copies up to n characters from the string pointed to by str2 to str1. Copying stops when n characters are copied or the terminating null character in str2 is reached. If the null character is reached, the null characters are continually copied to str1 until n characters have been copied. Returns the argument str1.

   Write a C program to implement and test iterative function
   **char** *strncpy_ite(**char** *str1, **const char** *str2, size_t n);

   **DO NOT** use any <string.h> functions in the the implementation of strncpy_ite() and **DO NOT** modify the main program. Program execution example:

   Example of program execution:



```
Tests of string copy with length n:

The library version:
  strncpy(str, "abc", 4) returns abc and str is abc.
  strncpy(str, "abcd", 4) returns abcd and str is abcd.
  strncpy(str, "abcde", 4) returns abcd and str is abcd.
  --------------------------------------------------
  strncpy_ite(str, "abc", 4) returns abc and str is abc.
  strncpy_ite(str, "abcd", 4) returns abcd and str is abcd.
  strncpy_ite(str, "abcde", 4) returns abcd and str is abcd.
```

(Continue to the next page)

2. (40 points) You may start with program skeleton **mexam_skeleton_2.c** and change the file name to **mexam_DXXXXXXX_2.c**. Let a[n] be an integer array of n elements. A version of insertion sort algorithm to sort array a[n] is given below:

```
 for (i=n-2; i>=0; i--)
         for (j=0; j<=i; j++)
                 if (a[i]>a[i+1]) swap(&a[i], &a[i+1]);
```

where swap(&x, &y) exchange the values of x and y. This question is to simulate the above algorithm to sort a singly-linked linear list. Define a node type Node and its pointer type NodePtr of an unordered singly-linked linear list as below:

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

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

Also, the type of a singly-linked linear list is defined as:

```
typedef NodePtr List; // type of a singly-linked linear list
```

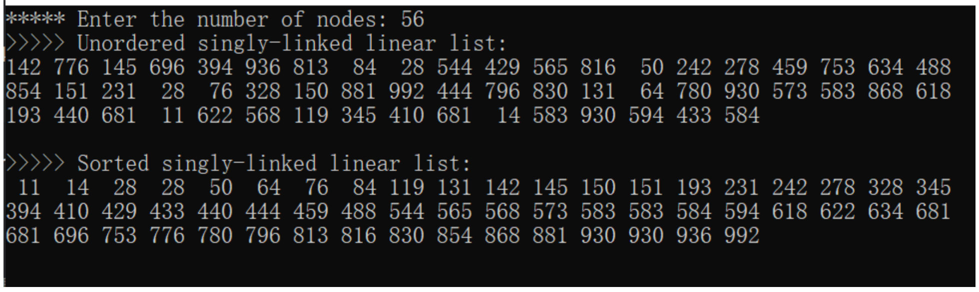
Suppose a node is inserted at the end of the list. That is, the node inserted at the earlier time is on the left hand side of the nodes inserted later. That is, the pointer of an unordered linear list points to the first inserted node on the most-left hand side of the list and the next link of the last inserted node points to NULL. Write a C program to implement the following functions:

1. **void** initial_list(List *L): Initialize list L to the empty list
2. **void** insert(List *L, **int** e): Insert a node of data element e to the end of list L.
3. **int** get_data(NodePtr ptr): Get the data of the node pointed by ptr.
4. **void** swap_node(NodePtr ptr1, NodePtr ptr2): Swap the data of two nodes pointed by ptr1 and ptr2.
5. **void** print_list(List L): Print list L from starting from the first node with 20 elements in a line.

In the main program perform the following steps:

1. Enter an integer n between 20 and 500 (including) to denote the number of the nodes of list lst.
2. Randomly generate an unordered duplicated singly-linked linear list lst with the elements between 0 and 999 (including).
3. Print the randomly generated list lst with 20 elements in a line.
4. Sort list lst into the ascending order by simulating the above sorting algorithm. Use **do-while** and/or **while-do** loop.
5. Print the sorted list lst with 20 elements in a line.

.Example of program execution:



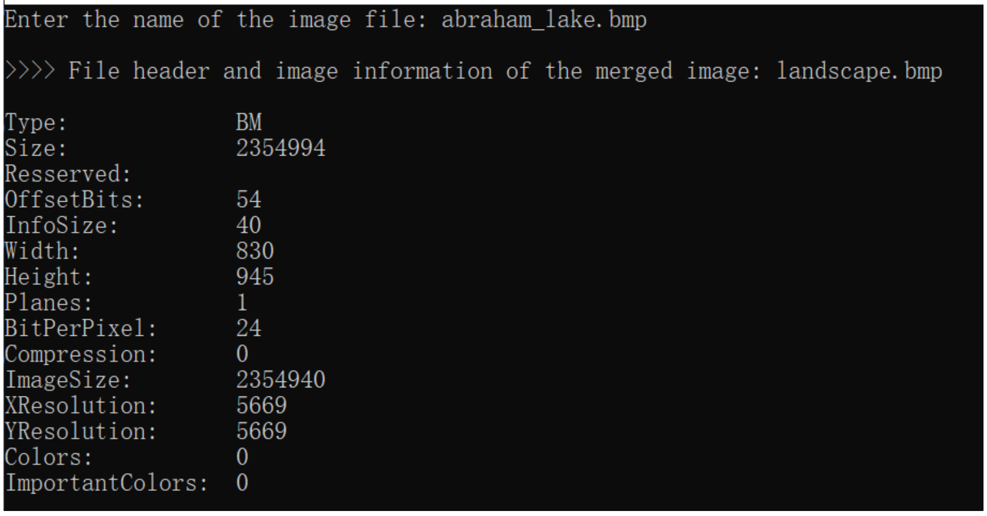
```
***** Enter the number of nodes: 56
>>>>> Unordered singly-linked linear list:
142 776 145 696 394 936 813  84  28 544 429 565 816  50 242 278 459 753 634 488
854 151 231  28  76 328 150 881 992 444 796 830 131  64 780 930 573 583 868 618
193 440 681  11 622 568 119 345 410 681  14 583 930 594 433 584

>>>>> Sorted singly-linked linear list:
 11  14  28  28  50  64  76  84 119 131 142 145 150 151 193 231 242 278 328 345
394 410 429 433 440 444 459 488 544 565 568 573 583 583 584 594 618 622 634 681
681 696 753 776 780 796 813 816 830 854 868 881 930 930 936 992
```

(continued to the next page)

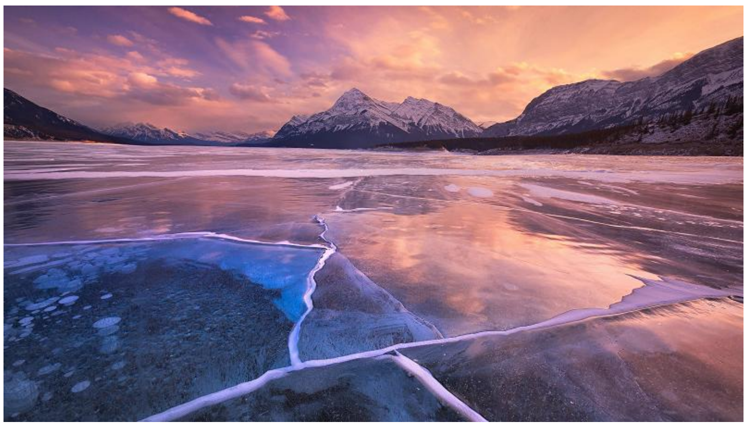
3. (40 points) You may start with program skeleton **mexam_skeleton_3.c** and change the file name to **mexam_DXXXXXXX_3.c**. An image is a ***portrait*** if its height is greater than width; otherwise, it is a ***landscape***. Let the frame of the resulting merged image be of width 15 pixels and with RGB color (20, 240, 240). Write a C program to perform framed image transformation as the following steps:
   1. Input the file name of a bitmap image and read the file to memory including file header, image information, color palette and image pixel data.
   2. Depending whether the image is a portrait or a landscape, perform the following image processing:
      2.1. If the image is a ***portrait***, 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 with a frame surrounding the merged image and a central vertical bar.
      2.2. If the image is a ***landscape***, perform transformation of mirror reflection along the horizontal line on the center of the image and attach the transformed on the bottom of the original image with a frame surrounding the merged image with a central horizontal bar.
   3. Output the merged image to disk using file name portrait.bmp or landscape.bmp for a portrait or a landscape of the original image, respectively.
   4. Print the file header and the image information of the merged image on the monitor.

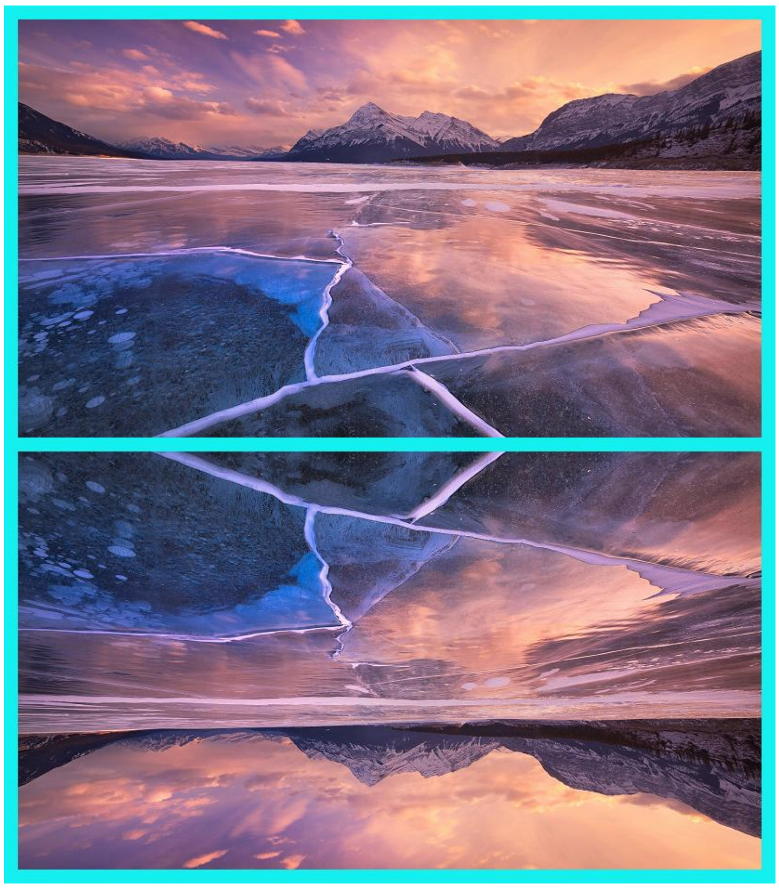
The data structure of the file header and the image information of bitmap images is given in the program skeleton. Use image abraham_lake.bmp and red_dragon.bmp to test the program. Examples of program execution: (in the next two pages)

The testing image abraham_lake.bmp (reduced size):

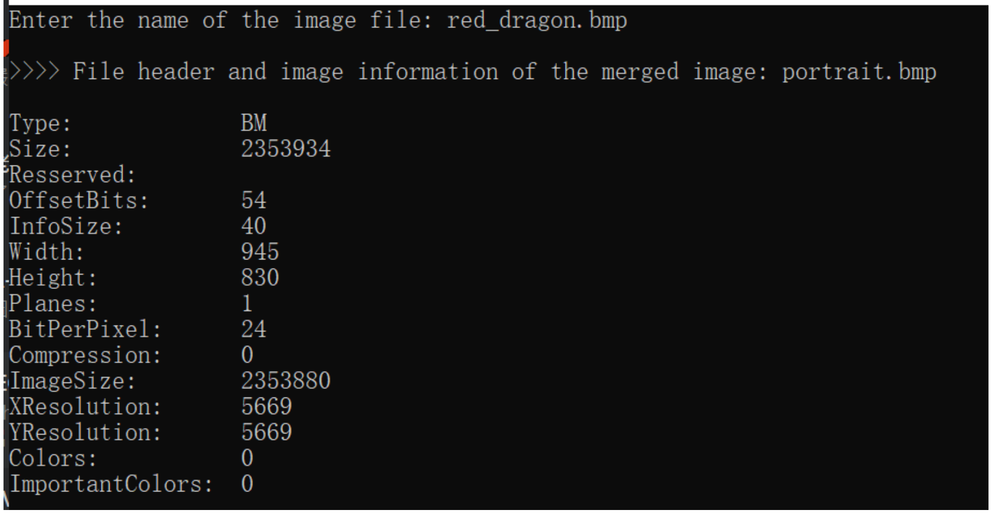


```
Enter the name of the image file: abraham_lake.bmp

>>>> File header and image information of the merged image: landscape.bmp

Type:             BM
Size:             2354994
Resserved:
OffsetBits:       54
InfoSize:         40
Width:            830
Height:           945
Planes:           1
BitPerPixel:      24
Compression:      0
ImageSize:        2354940
XResolution:      5669
YResolution:      5669
Colors:           0
ImportantColors:  0
```





The testing image red_dragone.bmp (reduced size):



```
Enter the name of the image file: red_dragon.bmp

>>>> File header and image information of the merged image: portrait.bmp

Type:             BM
Size:             2353934
Resserved:
OffsetBits:       54
InfoSize:         40
Width:            945
Height:           830
Planes:           1
BitPerPixel:      24
Compression:      0
ImageSize:        2353880
XResolution:      5669
YResolution:      5669
Colors:           0
ImportantColors:  0
```