Solution

```
#include <stdio.h>
enum {SUCCESS, FAIL, MAX_LEN = 80};
void BlockReadWrite(FILE *fin, FILE *fout);

main(void) {
        FILE *fptr1, *fptr2;
        char filename1[]= "lab1a.txt";
        char filename2[]= "lab1.txt";
        int reval = SUCCESS;

        if ((fptr1 = fopen(filename1, "w")) == NULL){
                printf("Cannot open %s.\n", filename1);
                reval = FAIL;
        } else if ((fptr2 = fopen(filename2, "r")) == NULL){
        printf("Cannot open %s.\n", filename2);
        reval = FAIL;
        } else {
         BlocReadWrite(fptr2, fptr1);
         fclose(fptr1);
         fclose(fptr2);
        }
        return reval;
}
```

Solution

```
void BlockReadWrite(FILE *fin, FILE *fout) {
        int num;
        char buff[MAX_LEN + 1];

        while (!feof(fin)){
         num = fread(buff, sizeof(char),
             MAX_LEN, fin);
         buff[num * sizeof(char)] = `\0';

         printf("%s", buff);
         fwrite(buff, sizeof(char), num, fout);
        }
}
```

Solution

```
#include <stdio.h>
enum {SUCCESS, FAIL, MAX_LEN = 80};
void BlockCat(FILE *fin);

main(int argc, char* argv[]) {
        FILE *fptr1, *fptr2;
        int reval = SUCCESS;
        if (argc !=2){
              printf("The correct syntax should be: cat1
              filename \n");
              reval = FAIL;
        }

        if ((fptr1 = fopen(argv[1], "r")) == NULL){
              printf("Cannot open %s.\n", argv[1]);
              reval = FAIL;
        } else {
         BlocCat(fptr1);
         fclose(fptr1);
        }
        return reval;
}
```

Solution

```
void BlockCat(FILE *fin) {
        int num;
        char buff[MAX_LEN + 1];

        while (!feof(fin)){
         num = fread(buff, sizeof(char),
              MAX_LEN, fin);
         buff[num * sizeof(char)] = `\0';

         printf("%s", buff);
        }
}
```

Exercise

- A)Improve the program in previous exercise so that it accepts the two filenames as command arguments.
- For example: if your program is named "filecpy". You can use it as the following syntax (in Linux):
- ./filecpy haiku.txt haiku2.txt
- B. Write a program having the same functionality as cat command in Linux
- ./cat1 haiku.txt

Hint

- Just use the argc[] et argv[]

```
if(argc<3) { printf("%s <file1> <file2>n",argv[0]); exit(1); }
```

- argv[1] and argv[2] will be the name of source file and destination file.

```
if((fp=fopen(argv[1],"r"))==NULL) {
…
};
if((fp2=fopen(argv[2],"w"))==NULL) {
…
};
```

Exercise: **Input and Output of structure**

- We assume that you make a mobile phone's address book.
- Define a data structure that can store "name," "telephone number," "e-mail address," and make an array of the structures that can hold at most 100 of the data.
- Input about 10 data to this array.
- Write a program to write the array content using fwrite() into the file for the number of data stored, and read the data into the array again using the fread ( ) function.

Solution

```
#include <stdio.h>

enum {SUCCESS, FAIL, MAX_ELEMENT = 20};

// the phone book structure
typedef struct phoneaddress_t {
        char name[20];
        char tel[11];
        char email[25];
}phoneaddress;

int main(void)
{
    FILE *fp;
    phoneaddress phonearr[MAX_ELEMENT];
     int i,n, irc; // return code
    int reval = SUCCESS;
```

Solution

```
   printf("How many contacts do you want to enter (<20)?");
    scanf("%d", &n);
   for (i=0; i<n; i++){
     printf("name:"); scanf("%s",phonearr[i].name);
     printf("tel:"); scanf("%s",phonearr[i].tel);
     printf("email:"); scanf("%s",phonearr[i].email);
   }
   if ((fp = fopen("phonebook.dat","w+b")) == NULL){
       printf("Can not open %s.\n", "phonebook.dat");
       reval = FAIL;
   }

   // write the entire array into the file
    irc = fwrite(phonearr, sizeof(phoneaddress), n, fp);
   printf(" fwrite return code = %d\n", irc);
   fclose(fp);
```

Solution

```
    //read from this file to array again
    if ((fp = fopen("phonebook.dat","rb")) == NULL){
       printf("Can not open %s.\n", "phonebook.dat");
       reval = FAIL;
    }
    irc = fread(phonearr, sizeof(phoneaddress), n, fp);
    printf(" fread return code = %d\n", irc);
    for (i=0; i<n; i++){
      printf("%s-",phonearr[i].name);
      printf("%s-",phonearr[i].tel);
      printf("%s\n",phonearr[i].email);
    }
    fclose(fp);
    return reval;
}
```

File Random Accessing

- Two functions: fseek() and ftell()
- fseek(): function to move the file position indicator to the spot you want to access in a file.
- Syntax

```
fseek(FILE *stream, long offset, int whence);
```

- Stream is the file pointer associated with an opened file
- Offset indicates the number of bytes from a fixed position
- Whence: SEEK_SET, SEEK_CUR, and SEEK_END
  - SEEK_SET: from the beginning of the file
  - SEEK_CUR: from the current position
  - SEEK_END: from the end of file

File Random Accessing

- ftell: obtain the value of the current file position indicator
- Syntax:

  long ftell(FILE *stream);

- rewind(): reset the file position indicator and put it at the beginning of a file
- Syntax:
- void rewind(FILE *stream);

Dynamic memory allocation

- Write a program to load a specific portion of the address book data from the file (for example, “3rd data to 6th data” or “2nd data to 3rd data”), modify something on the data, and finally save the data to the file again.
- But, you must allocate necessary minimum memory (the necessary size for “3rd data to 6th data” is four, while two for “1st data to 2nd data”) to save the data by the malloc( ) function.

Solution

```
#include <stdio.h>
#include <stdlib.h>

enum {SUCCESS, FAIL, MAX_ELEMENT = 20};

// the phone book structure
typedef struct phoneaddress {
        char name[20];
        char tel[11];
        char email[25];
}phoneaddress;

int main(void)
{
    FILE *fp;
    phoneaddress *phonearr;


    int i,n, irc; // return code
    int reval = SUCCESS;
    printf("Read from 2sd data to 3rd data \n");
```