CS 250
Homework 5-2

1. Write a C function with the following signature:

   ```
   int *ia_copy (int a[], unsigned int size)
   ```

   Your function should create a new array of the indicated size, initialize it using the data in `a`, and then return a pointer to the new array.

2. A program contains the following declarations:

   ```
   char customer_name[N_CUSTOMERS][MAX_NAME_LENGTH];
   int customer_number[N_CUSTOMERS]
   ```

   A program uses a text file that contains the following data on each line:

   ```
   <customer number> <customer last name> <customer first name>
   ```

   The customer number is an `int`, and the first and last names are alphabetic strings that contain no whitespace. The last and first names themselves are however separated by whitespace.

   Write a C function with the following prototype:

   ```
   void read_customer (char name[][MAX_NAME_LENGTH], int number[],
                       int position, FILE *cust_file)
   ```

   Your function[1] should read a single customer number and the last and first names from the file. Read the information using `fscanf()`. Store the information in the specified position in the arrays. The name should be stored in the array in the form <first><last>, with a space separating the two names. For example, if the file contains

   ```
   1234 Smith Suzie
   ```

   store the string "Suzie Smith" in the name array. (Hint: use `strcat()`.) Do not worry about buffer overflow in this problem: assume that $0 \leq$ `position` $<$ `N_CUSTOMERS`, and that any strings you manipulate have a length less than `MAX_NAME_LENGTH`.

3. Redo question 2, except assume the customer name array has been declared as

[1] When you pass a two dimensional array to a function, C requires you to specify the number of columns in the function declaration.

```
char *customer_name[N_CUSTOMERS];
```

so the function prototype will be

```
void read_customer (char *name[], int number[],
                    int position, FILE *cust_file)
```

Your function will need to dynamically allocate memory for the new string, and store a pointer to that memory in the appropriate place in the array. Make sure that the new string is no longer than it needs to be to contain the required characters.

4. Write a C program that will sum all of the arguments on the command line of the program and print out this sum. Assume that all of the arguments except `argv[0]` are properly formatted `ints`. Use `sscanf()` to convert the strings to `ints`. Here is an example of what your program should look like when it executes:

```
% ./sum 1 2 3 4
10
```

5. Write a C function with the following signature:

```
unsigned int remove_min (char *s_list[], unsigned int size)
```

The parameter `size` is the number of elements currently stored in the array. Your function should find the lexicographically-least string pointed to by an element of the array `s_list`. Free the memory associated with the string, and then compact the array by moving all of the strings that are after the position of the least string one position closer to the beginning of the array. Return the new size of the array as the value of the function.