

## Assignment 2

### Due Monday, April 28

This assignment uses pointers and arrays. The key concepts that will be covered in this assignment are interchanging the subscript operator (`[]`) with pointers, dynamically allocating an array, and passing parameters. Please remember, you must use pointers instead of the subscript operator. The only time the subscript operator should appear in your program is when you declare an array. Also note that you are not allowed to use C++ reference parameters. You must use pointers to simulate pass-by-reference parameters.

You will create an array of integers and keep track of the size and the number of items. *size* is defined to be the maximum number of items that can be stored in the array. *numitems* is the number of items that are currently stored in the array. At the start of the program, the user will be prompted for the size of the array. The array will then be dynamically allocated and the *numitems* should be initialized to zero.

The user should then be able to choose from a set of options that manipulate the array. A menu should display the options to the user and should allow the user to continue to choose options until the user specifies completion of the program. The options to be given are the following (each option is to be implemented with its own function):

- e** This function should enter a value into the array. It takes three parameters, a pointer to the array, the size, and the number of items. This should prompt the user for the value they wish to enter into the array. The first position of the array should be filled first, then the second, and so on. Give an error message if there is an attempt to store a value to a full array.

```
void enter(int *ip, int size, int *numitems);
```

- p** This function should print out the items in the array. It takes two parameters, a pointer to the array and the number of items in the array. It should print out the items in the array in a nicely formatted fashion.

```
void print(int *ip, int numitems);
```

- r** This function reverses the items in the array. It requires two parameters, a pointer to the array and the number of items in the array. The items in the array should be moved, such that they end up stored in the reverse order.

```
void reverse(int *ip, int numitems);
```

**f** This function should read numbers from an input file to be stored into the array. The function should take three parameters, a pointer to the integer array, the size, and the number of items in the array. Give an error message if there is an attempt to store a value to a full array. This means that all the integers in the input file may not be entered into to array. Enter integers until either the array is full (print out a message) or all have been read. The user should be prompted for the name of the input file. There is no format for the numbers that appear in the input file. They can be on a line by itself, many on one line, and blank lines can appear anywhere in the file.

```
void fileinput(int *ip, int size, int *numitems);
```

**q** Quit the program.

You should use pointers as indexes into the arrays. You are not allowed to use the subscript operator `[]`. You must have the declarations for the integer array, the size, and the number of items in the main program.