

CSE1310 INLAB 10

Note: No skeletal code will be provided for this lab. Refer to the previous labs (labs 7 through 9) to get the outline of the program.

Make sure to save source code for each checkpoint into a separate file. The naming convention of the files should be 'arrays-chk1.c', 'arrays-chk2.c', etc. For example, after you have completed checkpoint1, then copy 'arrays-chk1.c' to 'arrays-chk2.c' using the Unix copy command.

For this lab, you will write a C program that is menu-driven and uses the 'switch' statement with four cases providing the following four functionalities:

1. The first case will sort the elements of a given integer array in ascending order and display the contents of the resulting array.
2. The second case will sort the elements of the integer array in descending order and display the resulting array.
3. The third case will provide functionality to determine the length of a string.
4. The fourth case will determine if a given string is a palindrome or not. A palindrome is a word that reads the same even after it is reversed, e.g. the string 'ababbaba' is a palindrome.
5. The fifth case will be to add two matrices using arrays.

The 'switch' statement has to be enclosed within a 'while' loop that displays the menu of choices after the completion of a case. The 'while' loop conditional expression should cause the program to exit whenever the user wishes to (include a sixth case in your menu that prompts the user to exit the program) and also if an invalid input is entered other than the given choices.

Log into omega and using the 'vi' editor, create a file called 'arrays-chk1.c'. Implement following:

- Declare an integer array and a character array in the main function containing 10 integers and 9 characters respectively.
- Write two loops containing printf and scanf statements. First loop will be used to read the values into the integer array while second loop will be used to read the values into the character array.
- Write a while loop, which contains the print statements for displaying the menu. Look at the sample output at the end of the lab.
- Write the 'switch' statement after the print statements such that there is a function call to the corresponding functions: sort_ascend(), sort_descend(), str_length(), palindrome() or matrix_addition() in each case.
- Inside each function definition, provide printf statements to indicate which of the specific cases and functions have been selected. At this stage, your functions will be stubs, i.e. code that does no useful processing, but just prints out the name of the function.
- When you compile and execute the program, it should be able to print out a statement that displays the function name that the user entered, i.e., if the user selects choice 1, then the program should display a statement such as "The function sort_ascend() has been selected.". Signal the grader when you've finished this checkpoint.

Checkpoint 1

In this checkpoint, you'll work on the function `sort_ascend()`. This function is called in the first case of the 'switch' statement. You will pass an integer array to this function in order to sort the array. Use the following code to implement the sorting of the array:

```
void sort_ascend (int A[])
{
    int i, j, temp;
    for(i=0;i<10;i++)
    {
        for(j=9; j>i; j--)
        {
            if(A[j] < A[j-1])
            {
                temp = A[j];
                A[j] = A[j-1];
                A[j-1] = temp;
            }
        }
    }
}
```

After you write the above code, write one more loop to print out the contents of the sorted array. Signal the grader when you have completed this checkpoint.

Checkpoint 2

In this checkpoint, you'll work on the function `sort_descend()`. This function is called from the second case of the 'switch' statement. Similar to Checkpoint 2, this checkpoint will pass the same integer array to this function, but it will sort the numbers in descending order. Provide code that arranges the array elements in descending order. You can use the code of checkpoint 1 but make the necessary modifications to this code so that the array is sorted in descending order. Finally print out the sorted contents of the integer array. Signal the grader when you have an error-free, working program.

Checkpoint 3

In this checkpoint, you will write the code in function '`str_length()`' to calculate the length of a string. Pass the character array declared in the main function to this function. Implement following steps to find out the length:

1. Declare an integer `i` of type `int` and initialize it to 0.
2. Use a while loop to find out the length of the string. The condition in the while loop should be following:

```
str[i] != '\0'
```

Here str is name of the array received by this function. Basic approach is to search for the NULL ('\0') character in the character array. This NULL character signifies the end of the array.

3. Increment value of i within the loop
4. Print the value of i after the loop.

Signal the grader once you have a working error-free program.

Checkpoint 4

Here, you'll be pass the character array of main function as input to function 'palindrome'. In this function, you need to do the following steps:

- Use strlen() function of string.h header file to find out the length of the string. Store this length in an integer variable as given below:

```
len = strlen(str);
```

- Find out if the given string is palindrome by using the following loop:

```
for(i=len - 1, j=0; i>j; i--, j++)
{
    if(str[i] != str[j])
    {
        flag = 1;
        break;
    }
}
```

Here i, j, len and flag are variables of type int, and str is name of the array received by this function. Initialize the value of flag to 0.

- Check the value of flag after the loop. If it is zero, print that the string is a palindrome, else print that it is not palindrome.

Signal the grader when you have your program working without any errors.

Checkpoint 5

Here (default case), you will provide code that exits the program when none of the cases work. Use a printf statement to indicate that control is being transferred out of the program. (See the sample output). Make a call to the exit(0) function (built-in C function) to stop execution of the program at this point. This explicitly tells the program to stop execution beyond that point and control is transferred out of the program. Make sure you include the 'stdlib.h' C header file which defines the exit() function. Signal the grader when you've completed this checkpoint and your program is working after compilation and execution. You will work on matrix_addition() function in the next inlab(inlab 11) so keep your code and don't delete it.

Checkpoint 6

Sample output

Enter the elements of the integer array: 20 15 10 12 7 30 3 6 9 45

Enter the character string of maximum of 9 characters: badab

Choose one of the following

1. Sort in ascending order
2. Sort in descending order
3. String Length
4. Palindrome
5. Matrix Addition
6. Exit

1

The sorted array is 3 6 7 9 10 12 15 20 30 45.

Choose one of the following

1. Sort in ascending order
2. Sort in descending order
3. String Length
4. Palindrome
5. Matrix Addition
6. Exit

2

The sorted array is 45 30 20 15 12 10 9 7 6 3

Choose one of the following

1. Sort in ascending order
2. Sort in descending order
3. String Length
4. Palindrome
5. Matrix Addition
6. Exit

3

Length of the string is 5

Choose one of the following

1. Sort in ascending order
2. Sort in descending order
3. String Length
4. Palindrome
5. Matrix Addition
6. Exit

4

The given string is a palindrome.

Choose one of the following

1. Sort in ascending order
2. Sort in descending order

3. String Length
 4. Palindrome
 5. Matrix Addition
 6. Exit
- 6
Exiting the program.