Carnegie Mellon University Africa

Data Structures and Algorithms for Engineers

Practice Exercise 1

1) Set up development environment.

   - Install software development tools. It is recommended you setup Visual Studio Code.
   - Set up the DSA4E repository, e.g. C:\DSA4E
   - Create, compile and run a new program in C:\DSA4E\assignments\assignment1\myandrewid

2) You will learn how to open and close files for reading and writing. Thereafter, you will read and write one or more integer numbers.

   To read and write to a file, you can use the following.

```
int number_of_test_cases;
int i;
float t, x, y;

 …

/* read the number of test cases from a file */

 fscanf(fp_in, "%d", &number_of_test_cases); //note the &

 /* write the test case number to file */

 i = 1;
 fprintf(fp_out,"Test case %d\n",i); // print test case number


/* read the location data from a file */

 fscanf(fp_in, "%f %f %f", &t, &x, &y); //note the &

/* echo the input data to the screen */

 printf("Location data: %6.3f %6.3f %6.3f\n", t, x, y);
```

   Write the application code to read the input file for Assignment 1 and print the values to the terminal (i.e. echo them to the screen). You will then be sure you are reading valid data (and subsequently passing valid data to the function that implements your algorithm).

   To do this, you need to create an input file (input.txt) for Assignment 1 in the data directory and you need to copy to it the sample input data from the assignment handout.

3) Complete the application code you wrote in Question 2 in, e.g., `assignment1App.cpp`, as follows.

   Read and store the number of test cases, e.g. `number_of_test_cases`

   Add two nested loops:

   a) an outer `for` loop to iterate through each test case

b) an inner `while` loop to read the data for each test case.

In the inner loop, assign the three numbers associated with each location to three variables, e.g. `t, x, y,` terminating when you encounter `t` with the value less than zero.

4) Now, add code to write the following to the output file:

- your Andrew id at the beginning
- the time `t` and the `x` and `y` coordinates in the inner loop
- the ********** line after each test case, i.e. after exiting the inner loop.

5) After the data for each location is read, add a call to a function, e.g.

```
store_location(t, x, y); // NB x and y are integer values
```

6) Add the declaration of this function to the interface file, e.g., `assignment1Interface.cpp`

```
int store_location(float t, int x, int y); // NB x and y are integer values
```

7) Add the definition of this function to the implementation file, e.g., `assignment1Implementation.cpp`

```
int store_location(float t, int x, int y){

   /* this is just a stub */
   printf("store_location: time %6.3f (%4d, %4d) \n", t, x, y);

   return(0); // eventually, return a unique location id. number
};
```

8) Change the calling function in the inner loop so that it prints the value returned, e.g.

```
id = store_location(t, x, y);
printf("%3d    %6.3f %4d %4d\n", id, t, x, y);
```

9) After the lab, continue to develop your assignment as follows:

- Implement the data structure you have decided to use in the main program and pass it as an argument to `store_location()` so that the location point can be stored in the data structure.
- Alternatively, implement the data structure as a persistent static local variable in `store_location()`.
- Add functionality to `store_location()` so that, instead of just returning zero, it assigns and returns a unique identification number for that location.
- Add functionality to the main application so that it only writes a location to the output file if it has not been encountered previously.