

## COSC1137 - Fundamentals of Programming Lab II

### Lab 6

Fall 2022

This lab addresses the following major topics:

- Pointers
- Dynamic memory allocation: 1d array

You will find the instruction file along with any additional files on blackboard, when the lab starts.

In this category of lab activity students are expected to write a complete program during the lab and submit the compiling program on blackboard before the end of the lab session.

The instructor will give a brief description of the assignment and provides some (non-substantial) help throughout the lab. Students can use the textbook, class notes or sample programs to complete the assignment.

A research project conducts a study on people aged in the range 10 - 35 inclusive. It aims different studies on three age subcategories specified as follows:

Team A: the age range of 10 - 17  
Team B: the age range of 18 - 24  
Team C: the age range of 25 - 35

Students were notified about this project and were asked to volunteer themselves or their friends. They are asked to give their first names (one-word) and their ages. The important condition is that we are looking for one person of one specific age: no duplicate ages; so only one 15-year-old, one 26 year-old, etc.

In an input file called `applicants.txt` we have these names and ages, one person per line:

```
Mary 12
Jack 22
. . .
```

Since we know that the maximum age criteria is 35 years old, we declare `names` to be an array of size 36 and of type `string`. We initialize this `string` array to empty `strings` to start with.

Next we read from this file two pieces of information per line: first the name and next the age. The smart trick we would like to play here is to store the name of the person in the index that matches their age. So in our example above, `Mary` will be stored in index 12 in our array `names`, `Jack` will be in index 22, etc. And since we did not allow duplicates, we won't have the problem of having to store multiple 22 year olds in the same array element.

We do not know how many people signed up for this study, but we know their ages fit the criteria: 10-35 inclusive. So code based on this "lack of knowledge" about the number of lines in the input file. But you can keep the count of the applicants as you read through the file.

Just to see how this simple trick has an added bonus at this point of your program, go ahead and display your array `names`. Make sure to display the indices as well, since we know each name's index is indeed their age.

Did you expect the pattern that you see?

Did you expect to see the empty `strings`? What do they represent? If you have any questions about this, ask the lab assistant.

Next step for us is to populate the three teams. Not knowing how many persons fit in each of those subcategories, we would like to use pointers and dynamic memory allocation for each array representing the target age subcategories.

So, we start by counting the number of people who fit in each team's age criteria. Based on this count we can dynamically allocate a `string` array and store the names of the applicants in that age range. This way our age-based teams are properly stored in three dynamically allocated `string` arrays.

In order to populate each of these dynamic arrays, YOU MUST use `names` array.

Name these arrays `teamA`, `teamB`, and `teamC`. Remember these are our dynamically allocated `string` arrays storing the names of each team's members.

Before getting to the menu, you need to create the following functions to use in this assignment. The prototypes help you create and use these functions where applicable:

```
void displayTeam ( string *, int );  
string * merge ( string *, int, string *, int);
```

The `merge` function addresses the menu option 5. As you see this function receives two `string` arrays along with their respective sizes, merges them into a local dynamic array that it will return after populating it as a result of merging. The process of merging will be similar to appending one list at the end of other.

Now you are ready to display the following menu and let the user test your code:

Please choose an option:

1. Display the names of the members of every team on separate lines
2. Display the ages that are not represented in this research
3. Display the names of the youngest and the oldest participants regardless of teams
4. Display the names and ages of the youngest volunteers on each team.
5. Merge two teams A and B and display the merged team's members
6. Exit

Important notes on the menu options:

**Option 1.** You MUST call `displayTeam` function for this option.

**Option 2.** You need to use the `names` array for this option.

**Option 3.** What do you think about this option? Do you need to call the `youngestPerson` function? Do you need it? Think carefully before coming up with a solution for this option. It might be way easier than you thought!

**Option 4.** Must call the `youngestPerson` function for this option.

**Option 5.** MUST call the `merge` function, receive the pointer/dynamic array it returns and pass that merged array to your `displayTeam` function.

- *Submit your program by the end of the lab session you attended in person. Only submissions by students who attended the lab in person will be considered for grading.*
- *Make sure you submit a program that compiles. An incomplete program that compiles could possibly earn partial points, a non-compiling program will be considered void.*
- *This is an individual work, you may ask the instructor for some help. However you should not interact with fellow students.*