Assignment 1: Documentation

This was really fun to implement. I had to search down how I could iterate through an unordered_map in C++, I had to learn how buffers work, I had to make it look simple and nice to look at in code.

Data structures used:
- Student structure to house name, id, age, grade
- Hashmap for student container

This is what it looks like in the terminal when you first run it:

```
dowiw@DOILAPTOP:~/UE_Parallel_Programming_2024/AS01$ ./main
---Student Management System---
  1: to add student: needs name, id, age, grade
  2: to display all students.
  3: to find student by ID
  4: to delete student by ID
  5: calculate average grade of students
  6: to calculate factorial of a number
  7: to exit
  Enter an option from above.
------------------------------
```

If you **enter '1**' you will be asked to input the details of the student:

```
1
----------
Enter student name(string): KB
Enter student ID(int): 1
Enter student age(int): 1
Enter student grade(float): 98.7
Student successfully added.
----------
```

```
1
----------
Enter student name(string): KB
Enter student ID(int): 1
Enter student age(int): 19
Enter student grade(float): 98.4
----------
Student ID already exists.
----------
```

It will store this new student structure in its respective ID in the hashmap.
It also checks if the student ID already exists.

Note: for some reason I couldn't find a solution on how I could check if the user inputs a wrong type in the int(ID, age) and float(grade) that uses cin >>. It causes it to go on an infinite loop sadly.

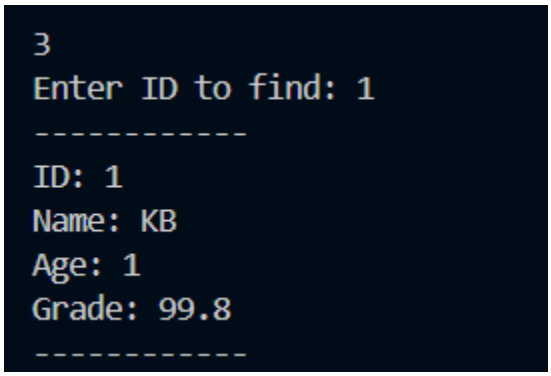
Option 2 will display all students created thus far. It will also check for if it is empty.

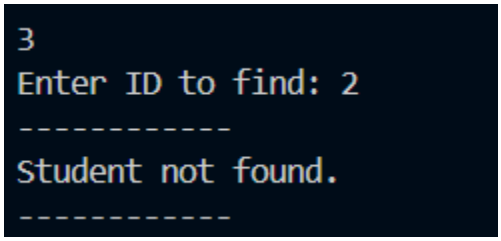
```
---Student Management System---
  1: to add student: needs name, id, age, grade
  2: to display all students.
  3: to find student by ID
  4: to delete student by ID
  5: calculate average grade of students
  6: to calculate factorial of a number
  7: to exit
  Enter an option from above.
------------------------------
2
------------
ID: 19
Name: KB
Age: 19
Grade: 99.8
------------
```

```
2
Registry is empty. Please input students.
```

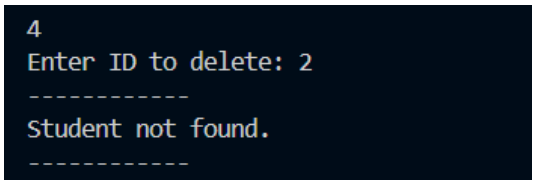
Assignment 1: Documentation

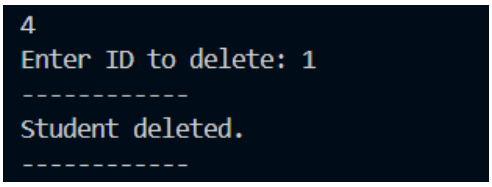
Option 3 will find the student using an ID provided. It will also return when a student is not found.



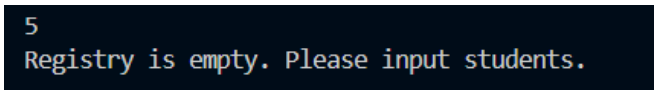


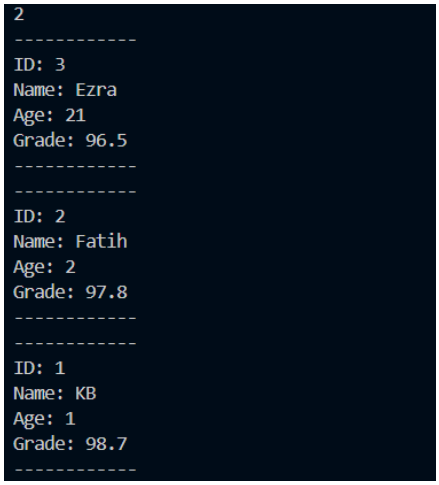
Option 4 has the same logic, but this time it uses the empty() method to delete the student using an ID.

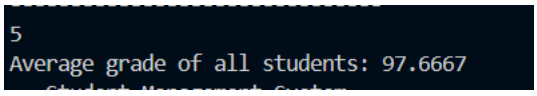




Option 5 calculates the floating average of the grades of all students. There is an iterator beneath it and divides it by the size of the map.
It also checks if there are no students at all.

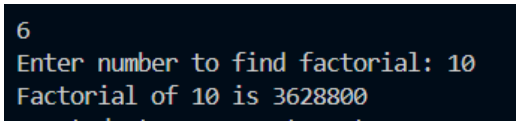






Assignment 1: Documentation

Option 6 finds the factorial of a number recursively.



Option 7 exits the program. This makes the buffer false so that the while-loop can end.

It definitely needs some work with the type checking but it is way too complicated for me. **Users** will have to be careful about what they input to avoid an infinite loop sadly.

What I learned:

- I learned a lot about how buffers work. How I could utilize a boolean to make a program run forever until it is turned false.
- I learned that I could pass through the memory address of a hashmap in the parameters of a function so that I could use it throughout the program. This makes sure that I don't make a new one again and again.
- I learned that working with structures as a hashmap value is very complicated. I had to look for a solution to iterate through the hashmap.
- This made me learn "auto" which is an dynamic iterator data-type that will forcibly look for the data type of what you want to have. In stack overflow, it was said that because hashmaps have keys and values, auto is a good way of getting the "type" of the "keys and values". Really weird, but I had a lot of fun using it.
- I also learned that I could use auto to get into the structures more easily using structural binding. Props again to stack overflow.
- Overall, a fun project to implement. It was challenging, took me a lot of time to get the logic done, but I liked it. I may also have an advantage because I like C a lot.