NO: TAN Y. IM DATE: 7/1/2022

Tutorial 2

~~1 a)~~ - Your grade is A.
- Your grade is B.
- Your grade is C.
- Your grade is D.
- Your grade is F.
- That is an invalid score. Run the program
~~b)~~ again and enter a value in the range of
0 through 100.

1 a) - Enter your numeric test score and I will
tell you the letter grade you earned: 80
Output Your grade is B.

- Enter your numeric test score and I will
tell you the letter grade you earned: 55
Output Your grade is F.

- Enter your numeric test score and I will
tell you the letter grade you earned: A
Output That is an invalid ~~score~~ score. Run the program
again and enter a value in the range of
0 through 100.

- Enter your numeric test score and I will
tell you the letter grade you earned 200
Output That is an invalid score. Run the program
again and enter a value in the range of
0 through 100.

explain how constant ad affect variable the prog

b) Area 1 is declaring variables in constant integer which named A_SCORE, B_SCORE, C_SCORE, D_SCORE, MIN_SCORE, and MAX_SCORE.

What about asking user input

Area 2 is ask user to enter the ~~marks~~, ~~thus~~ ~~the~~ testScore (marks), thus the testScore is determine ~~that~~ ~~whetel~~ whether is bigger or equal to the ~~MINSCO~~ MIN_SCORE and smaller or equal to MAX_SCORE Next, the letter grade is ~~test~~ determine in the nested if-else statement.

Area 3 is when the testScore is not bigger or equal to MIN_SCORE and testScore is not smaller or equal to MAX_SCORE. Thus, the ~~out p~~ output will show "That is an ~~that~~ invalid ~~outp~~ score. Run the program again and enter a value in the range of MIN_SCORE through MAX_SCORE."

What about out of range value

2.

num 1	num 2	num 3	avg.
-	-	-	-
-	-	-	-
-	-	-	—
3	-	-	-
3	-	-	-
3	3	-	-
3	3	-	-
3	3	3	-
3	3	3	-
3	3	3	3

Pls see me

3. Acceptance testing. Verifying whether the whole system works as intended.
~~th~~ Integration testing. Ensuring that software components or functions operate together
Unit testing. Validating that each software unit performs as expected. A unit is the smallest testable component of an application.
Functional testing. the functions are checked by emulating business scenarios, based on functional requirements.
Performance testing. How the software performs is tested under different workloads.
Regression testing. Checking whether new features break or degrade functionality.
Stress ~~tes~~ testing. Testing how much strain the system can take before it fails.
Usability testing. How well a customer can use the system to complete a task is validate.

4. - Functional defects. the errors identified in case the behavior of sofware is not compliant with the functional requirements.
. Performance defects.. The bound to software's speed, stability, ~~responset~~ response time, and resource consumption, and ~~t~~ are discovered during performance testing.
- Usability defects. A content layout that is difficult to scan or navigate and an overly complex signup procedure.
- Compatibility defects. An application with compatibility errors doesn't show consistent performance on particular types of hardware, operating systems, browsers, and devices or when integrated with certain software or operating under certain network ~~t~~ configurations.
. Security defects. Encryption errors, susceptibility to SQL injections, XSS vulnerabilities, buffer overflows, weak authentication, and logical errors in role-based access, are the most frequent security defects.
. Critical defects. An entire system's or module's functionality is blocked, and ~~it~~ testing cannot proceed further without such a defect being fixed.
- High-severity defects. key functionality of ~~an applicati~~ application, and the app behaves in a way that is strongly different from one stated in the requirements.
- Medium-severity defects. A minor function does not behave in a way stated in the requirements.
. Low-severity defects. An applications UI and may include such an example as a slightly different size of a button.

good.

NO: TAN YI JIA DATE: 7/9/2022

lab ~~#2~~ exercise 2.

1. a) int empNums[100];
   b) float payRates[25];
   c) long miles[14];
   d) string cityName[26];

2. i) The size of the array should only be positive.
   ii) The size of the array should not be a float.
   iii) The array size is not specified
   iv) The "size" does not have value is not an array.

3. double s[4] = {19.22, 22.19, ~~&~~ 11.11, 12.11}

4. The size declarator is used in a definition of array to indicate the number of elements the array will have. Example: int age[5];
The subscript is used to access a specific element in an array.
Example: age[1] = 10.

~~5. i) The size declarator is used in a definition of an array to indicate the number Of elements. Eg: int num[2];~~

~~ii) A subscript is used to access a specific element in array. num[1] = 0;~~

5. C++ does not do array bounds checking automatically compared to Python.
Array bound ~~the~~ checking refers to determining whether all array references in a program are within the declared ranges.

6. output :

```
1
2
3
4
5
```

7.

```
# include<iostream>
using namespace std;
int main( )
{
  const int NUM_FISH = 20;
  int fish [NUM_FISH];

  for ( i=0; ~~ee~~ i<20; i++)
    {
      cout << "How many fish you were caught?  ";
      cin >> fish[i];
    }
```

NO:.............................. DATE:..............................

```
    for (i=0;  i< 20;  i++ )
        cout << fish[i] << endl;
    return 0;
}
```

8. NO. Array can't be assigned, but it can be copied using "copy(array1, array1+size, array2); considering having "using namespace std" instead of using "array1 = array2"

9 Depends. sometimes an address is being passed (pass by reference), sometimes pass by value (a copy is being made)

Good.

perhaps you can try to finish up the remainder quickly.