Lab 7 – Submit Task 7.3 and Task 7.4 as part of assignment 2.

Task 7.1 Find the error

Find the error in each of the following program segments and correct the error:

a)
```
#include <iostream>;
```

b)
```
arraySize = 10; // arraySize was declared const
```

c) Assume that
```
int b[ 10 ] = {};
for ( int i = 0; i <= 10; i++ )
      b[ i ] = 1;
```

d) Assume that
```
int a[ 2 ][ 2 ] = { { 1, 2 }, { 3, 4 } };
a[ 1, 1 ] = 5;
```

e)
```
double cube(float );   /*function prototype*/
...
cube(float number ) /*function prototype*/
{
      return number*number *number;
}
```

f)
```
double y =123.45678;
int x;
x = y;
cout<<(double)x;
```

g)
```
double square(double number )
{
double number;
```

©Copyright: 2019 Swinburne University of Technology

```
      return number *number;
      }
h)  double f[ 3 ] = { 1.1, 10.01, 100.001, 1000.0001 };
```

Task 7.2 Use a single-subscripted array to solve the following problem. Read in 20 numbers, each of which is between 0 and 99, inclusive. As each number is read, print the number and indicate 'duplicate number' if it is a duplicate number. After all 20 numbers have been entered; call a function to print out the most frequent number in the array.
[Sample solution is available in Canavs]

Task 7.3 (Need to submit as a part of assignment 2)

Create a one-dimensional array to read 20 alphabetical letters (your program should be able to detect and print out an error message if a non-alphabetical letter is entered). As each letter is entered, print a message saying 'duplicate letter' if the letter is already in the array. Write a function that can sort the array after all 20 letters have been entered. Write another function that print out the most frequent letter and number of times it was entered. Prepare for the case where all 20 letters are different, or all are the same. [Refer to sample solution given for Task 7.2]

Sample output:

```
Prince@Raji-PC /h/C-New/ISD/ISD_TP2_2017/week7
$ a
Enter 20 alphabets
a
b
a
a is a duplicate letter.
4
4 is not an alphabet
#
# is not an alphabet
d
d
d is a duplicate letter.
d
d is a duplicate letter.
d
d is a duplicate letter.
f
d
d is a duplicate letter.
d
d is a duplicate letter.
f
f is a duplicate letter.
g
h
r
d
d is a duplicate letter.
d
d is a duplicate letter.
r
r is a duplicate letter.
e
d
d is a duplicate letter.
f
f is a duplicate letter.

The original array is:
abaddddfddfghrddredf
The sorted array is:
aabddddddddefffghrr
The mode is 'd' and it is occuring 9 times.
```

©Copyright: 2019 Swinburne University of Technology

```
Prince@Raji-PC /h/C-New/TSD/TSD_TP2_2017/week7
$ a
Enter 20 alphabets
a
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.
a
a is a duplicate letter.

The original array is:
aaaaaaaaaaaaaaaaaaaa
The sorted array is:
aaaaaaaaaaaaaaaaaaaa
All characters are the same.
```

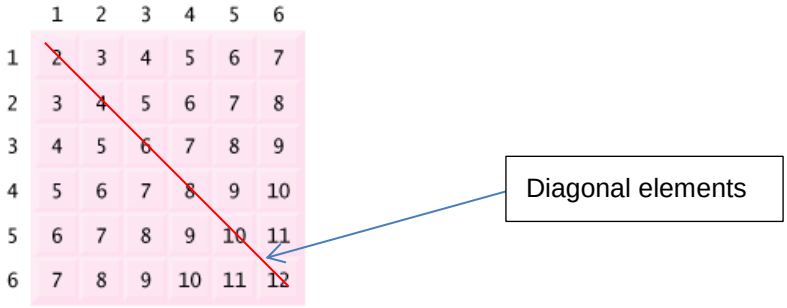
```
Prince@Raji-PC /h/C-New/TSD/TSD_TP2_2017/week7
$ a
Enter 20 alphabets
a
3
3 is not an alphabet
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t

The original array is:
abcdefghijklmnopqrst
The sorted array is:
abcdefghijklmnopqrst
All characters are entered only one time.
```

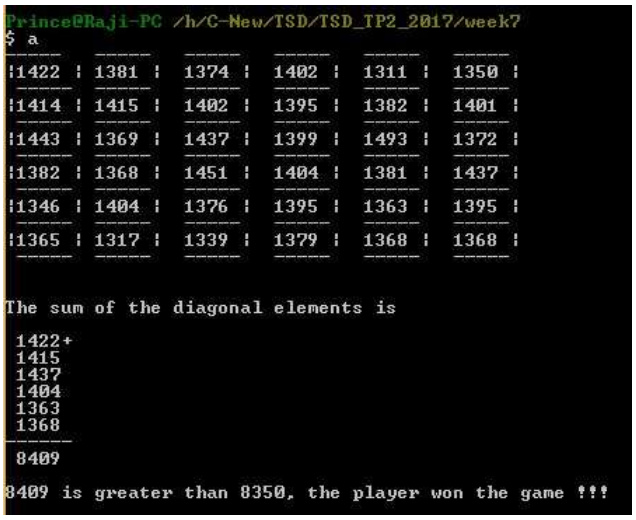
©Copyright: 2019 Swinburne University of Technology

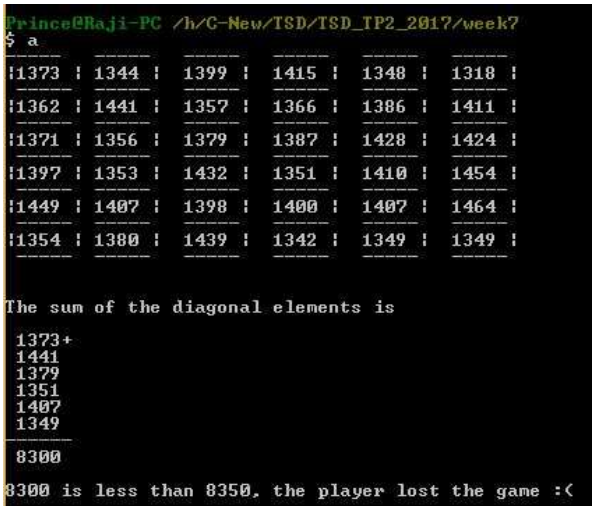
Task 7.4 (Need to submit as a part of assignment 2)

Write a program that simulates the rolling of two dice. The program should use **rand** to roll the first and second dice. The sum of the two values should then be calculated. [Note: Since each die can show an integer value from 1 to 6, then the sum of the two values will vary from 2 to 12, with 7 being the most frequent sum and 2 and 12 the least frequent sums.]. There are 36 possible combinations of the two dice. Your program should roll the two dice 50,000 times. Use a 2D array to tally the numbers of times each possible sum appears. Print the results in a tabular format. The player wins if the total of the diagonal elements is greater than 8350, otherwise the player loses.



Sample output: (The table formatting should look like the same as in the screen shots below)





©Copyright: 2019 Swinburne University of Technology