1 **Array and Vector**

Write a C++ program as follows:

Declare 2 empty vectors of integers named
vector1 and vector 2, respectively.

Add 10 and 20 to vector1 dynamically using push_back
Display the elements in vector1 using the at() method as well as its size using the size() method

Add 100 and 200 to vector2 dynamically using push_back
Display the elements in vector2 using the at() method as well as its size using the size() method

Declare an empty 2D vector called vector_2d
Remember, that a 2D vector is a vector of vector<int>

Add vector1 to vector_2d dynamically using push_back
Add vector2 to vector_2d dynamically using push_back

Display the elements in vector_2d using the at() method

Change vector1.at(0) to 1000

Display the elements in vector_2d again using the at() method

Display the elements in vector1

2 **Statements and Operators**

For this program I will be using US dollars and cents.

Write a program that asks the user to enter the following:

An integer representing the number of cents

You may assume that the number of cents entered is greater than or equal to zero

The program should then display how to provide that change to the user.

In the US:

1 dollar is 100 cents
1 quarter is 25 cents
1 dime is 10 cents
1 nickel is 5 cents, and
1 penny is 1 cent.

Here is a sample run:

Enter an amount in cents : 92

You can provide this change as follows:
dollars : 0
quarters : 3
dimes : 1

nickels : 1
pennies : 2

3 **Controlling Program**

This challenge is about using a collection (list) of integers and allowing the user to select options from a menu to perform operations on the list.

Your program should display a menu options to the user as follows:

P - Print numbers
A - Add a number
M - Display mean of the numbers
S - Display the smallest number
L - Display the largest number
Q - Quit

Enter your choice:

The program should only accept valid choices from the user, both upper and lowercase selections should be allowed.
If an illegal choice is made, you should display, "Unknown selection, please try again" and the menu options should be
displayed again.

If the user enters 'P' or 'p', you should display all of the elements (ints) in the list.
If the list is empty you should display "[] - the list is empty"
If the list is not empty then all the list element should be displayed inside square brackets separated by a space.
For example, [1 2 3 4 5]

If the user enters 'A' or 'a' then you should prompt the user for an integer to add to the list which you will add to the list and then display it was added. For example, if the user enters 5
You should display, "5 added".
Duplicate list entries are OK

If the user enters 'M' or 'm' you should calculate the mean or average of the elements in the list and display it.
If the list is empty you should display, "Unable to calculate the mean - no data"

If the user enters 'S' or 's' you should display the smallest element in the list.
For example, if the list contains [2 4 5 1], you should display, "The smallest number is 1"
If the list is empty you should display, "Unable to determine the smallest number - list is empty"

If the user enters 'L' or 'l' you should display the largest element in the list
For example, if the list contains [2 4 5 1], you should display, "The largest number is 5"
If the list is empty you should display, "Unable to determine the largest number - list is empty"

If the user enters 'Q' or 'q' then you should display 'Goodbye" and the program should terminate.

Before you begin. Write out the steps you need to take and decide in what order they should be done.

Think about what loops you should use as well as what you will use for your selection logic.

This exercise can be challenging! It may likely take a few attempts before you complete it -- that's OK!

Finally, be sure to test your program as you go and at the end.

Hint: Use a vector!

4 **Characters & strings**

A simple and very old method of sending secret messages is the substitution cipher.
You might have used this cipher with your friends when you were a kid.
Basically, each letter of the alphabet gets replaced by another letter of the alphabet.
For example, every 'a' get replaced with an 'X', and every 'b' gets replaced with a 'Z', etc.

Write a program thats ask a user to enter a secret message.

Encrypt this message using the substitution cipher and display the encrypted message.
Then decryped the encrypted message back to the original message.

You may use the 2 strings below for your subsitition.
For example, to encrypt you can replace the character at position n in alphabet
with the character at position n in key.

To decrypt you can replace the character at position n in key
with the character at position n in alphabet.

5 **Pointers & Refrence**

Write a C++ function named apply_all that expects two arrays of integers and their sizes and dynamically allocates a new array of integers whose size is the product of the 2 array sizes

The function should loop through the second array and multiplies each element across each element of array 1 and store the
product in the newly created array.

The function should return a pointer to the newly allocated array.

You can also write a print function that expects a pointer to an array of integers and its size and display the
elements in the array.

For example,

Below is the output from the following code which would be in main:

```
int array1[] {1,2,3,4,5};
int array2[] {10,20,30};

cout << "Array 1: " ;
print(array1,5);

cout << "Array 2: " ;
```

```
    print(array2,3);

    int *results = apply_all(array1, 5, array2, 3);
    cout << "Result: " ;
    print(results,15);

   Output
   ---------------------
   Array 1: [ 1 2 3 4 5 ]
   Array 2: [ 10 20 30 ]
   Result: [ 10 20 30 40 50 20 40 60 80 100 30 60 90 120 150 ]
```

6 **OOP – classes and objects**

For this challenge you are to develop the foundation for a program
for movie fanatics to keep track of what movies they have watched
and how many times they watched each movie.

The program must support the following:

class Movie - models a movie which includes
- movie name
- movie rating (G, PG, PG-13, R)
- watched - the number of times the movie has been watched

class Movies - models a collection of movie objects

Obviously, Movies needs to know about Movie since it is a collection of Movie object
However, our main driver should only interact with the Movies class.

For example. a simple main should be able to
- create a Movies object
- ask the Movies object to add a movie by providing the movie name, rating and watched count
- ask the Movies object to increment the watched count by 1 for a movie given its name
- ask the Movies object to display all of its movies

Additionally,
- if we try to add a movie whose name is already in the movies collection
we should display this error to the user
- if we try to increment the watched count for a movie whose name is not in the movies
collection we should display this error to the user

I've provided a basic shell as a starting point for one possible solution
that has fully implemented
- Movie and main

You can choose to use my starting point or start from scratch.

Here is what your project files should look like:
-Movie.h - include file with the Movie class specification
-Movie.cpp - file with the Movie class implementation
-Movies.h - include file with the Movies class specification
-Movies.cpp - file with the Movies class implementation

-main.cpp - the main driver that creates a Movies object and adds and increments movies

7 **Operator Overloading**

Overload the following operators in the provided Mystring class.

To gain experience overloading operators, you should do this challenge twice.
First, overload the operators using member functions and then in another project overload the same operators
again using non-member functions.

Please do it once using member methods and then again using non-member functions.

Here is a list of some of the operators that you can overload for this class:

```
-       - unary minus. Returns the lowercase version of the object's string
          -s1
==     - returns true if the two strings are equal
         (s1 == s2)
!=     - returns true if the two strings are not equal
         (s1 != s2)
<      - returns true if the lhs string is lexically less than the rhs string
         (s1 < s2)
>      - returns true if the lhs string is lexically greater than the rhs string
        (s1 > s2)
+       - concatenation. Returns an object that concatenates the lhs and rhs
         s1 + s2
+=     - concatenate the rhs string to the lhs string and store the result in lhs object
        s1 += s2;      equivalent to s1 = s1 + s2;
*       - repeat -  results in a string that is copied n times
        s2 * 3;         ex). s2 = "abc";
                            s1 = s2 * 3;
                            s1 will result in "abcabcabc"
*=      - repeat the string on the lhs n times and store the result back in the lhs object
         s1 = "abc";
         s1 *= 4;       s1 = s1 will result in "abcabcabcabc"
```

If you wish to overlod the ++ and -- operators remember that they have pre and post versions.

The semantics should be as follows (this shows the member method version):

```
Mystring &operator++()   {  // pre-increment
  // do what ever you want increment do to - maybe make all characters uppercase?
  return *this;
}

// Note that post-increment returns a Mystring by value not by reference
Mystring operator++(int) {   // post-increment
  Mystring temp (*this);       // make a copy
  operator++();                 // call pre-increment - make sure you call pre-increment!
  return temp;                 // return the old value
}
```

8 **Inheritance**

You are provided with a simple Account class hierarchy that contains
Account - Base class
Savings Account - Derived class

An Account has a name and a balance.
A Savings Account is an Account and adds an interest rate.

I have also provided some Account helper functions in Account_Util.h and Account_Util.cpp that make it easy to display, deposit to, and withdraw from a vector of Accounts and Savings Accounts.

Please refer to the source code provided and the video explanation of this challenge for more details.

Your challenge is the following:

1. Add a Checking account class to the Account hierarchy
   A Checking account has a name and a balance and has a fee of $1.50 per withdrawal transaction.
   Every withdrawal transaction will be assessed this flat fee.

2. Add a Trust account class to the Account hierarchy
   A Trust account has a name, a balance, and an interest rate
   The Trust account deposit works just like a savings account deposit.
   However, any deposits of $5000.00 or more will receive a $50.00 bonus deposited to the account.

   The Trust account withdrawal should only allow 3 withdrawals per year and each of these must be less than 20% of the account balance.
   You don't have to keep track of calendar time for this application, just make sure the 4th withdrawal fails :)