1. Write a program that declares an array of 5 elements and uses a loop to assign the values 1.1, 1.2, 1.3, 1.4, and 1.5 to them. Next, the program should display the array’s elements in reverse order.

2. Declare two arrays of characters, say, chArrayA and chArrayB. Include cstring header and use the functions availble for string manipulation. Use these two arrays to -
a) Read from console first name and copy it to chArrayA and read last name and copy it to chArrayB.
b) Copy last name to chArrayA after leaving one space in between first and last name. Display the name to console. If chArrayA had "Lavneet" and chArrayB had "Singh" after copying, chArrayA should've "Lavneet Singh".
c) Read a multi word string from console to chArrayA.
d) Copy chArrayA into chArrayB with all its characters in reverse order. Eg. if chArrayA has "abc", then after copying chArrayB should have "cba". Both chArrayA & chArrayB should be c-style strings
e) Copy chArrayA into chArrayB with words in reverse order. i.e. if chArrayA has "This is good" then chArrayB should have "good is This"
f) Read string from console into chArrayA and copy only its alphabets to chArrayB. Eg. if input string is "Hello! don't worry. Dial 101010 for all your queries" then chArrayB should be "HellodontworryDialforallyourqueries". Use cctype header, std::isalpha
g) In all of your programs in Q2 above, verify your code's resiliency by testing edge cases like,
- input only one char from console
- input empty string (by hitting enter key)
- input only one word string where multi-word string is expected
- input all non-alpha chars for 2(f)

Being able to handle edge cases properly is an important programming skill. Practice it and adapt habit of testing edge cases.

3. Write a program that a cinema's office can use for booking tickets. Suppose that the cinema has 16 rows with 20 seats each.
The program should display the seats status in 16x20 two dimensional tablular figure on console. Use arrays of characters for displaying the grid. Like:

```
Current Status: (if each row had 3 seats, eg):
* * X --> A
* X * --> B
* * * --> C
:
and so on
:
* * * --> P

vvvvvvvvvvv Screen this way.

* => available seat
X => occupied / unavailable seat
```

Below this display, the program should display a menu to perform the following operations:
1. Buy a ticket
2. Cancel ticket

3. Exit the app
The booking office should be able to book/cancel a seat by providing its row alphabet and a number as input. eg. If office person enters 1, and then A 15, it should show an X character against A-15 cell in the figure and mention "A-15 booked". If the seat is already taken, "Error: A-15 unavailable".
Once the user operation is done, the program should get back to it's 'home screen' with updated seat status display.

4. Let's try building an order handling software (OHS) for, say, a pizza house. The component that we are interested in deals with only order numbers, not the content of the orders.
The home screen should show following:

```
1. Place order
2. Pending orders
3. Serve order
Enter your pick:
```

All employees of the house are seeing the same screen. When an employee enters 1: it just adds an order number in the pending order list. It displays the order number added, like: "Your order is placed. Order Number: 1043". It then waits for user to hit enter and gets back to home screen after it is done.
When user enters 2: It displays the pending orders and their total count
When user enters 3: The program should display list of pending orders and print in next line: "Pick order:". The employee will enter an order number. If the order entered matches one of pending orders in the list, it deletes that order from the list and returns to home screen (simulating that the order has been served).
Note: You can pick whatever numbers you want to represent orders. Like, starting with 100 or 1000 or 2000 etc. Also, the pending orders list should've no empty cells in between orders. 2nd, Option 3 can be used to pick any of the pending orders.

5. Create a two dimensional square array decor[MAX][MAX]. Fill this array up with two types of charaters, '-' and '*'. The fill should be such that when this array is displayed as a grid, following pattern is seen:
Outermost boder of the box/table is filled of '-' characters. Then the inner border is filled of '*' charaters. Then the next inner border of '-' and so on till no inner border is left to be filled. Take MAX as 11 to begin with. For MAX as 11, it should look like:

```
-----------
-*********-
-*-------*-
-*-*****-*-
-*-*---*-*-
-*-*-*-*-*-
-*-*---*-*-
-*-*****-*-
-*-------*-
-*********-
-----------
```

NOTE:

In the programs above, don't write program that display directly to console. Meaning, take example of Q5. Instead of writing code to display the pattern directly to screen, you should fill the pattern into decor[][] array first. And, then you a separate piece of code simply displays the content of the array. This will help you practice arrays better.