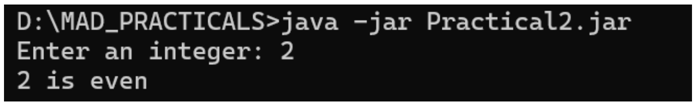
Mobile App Development Practical Sheet 2

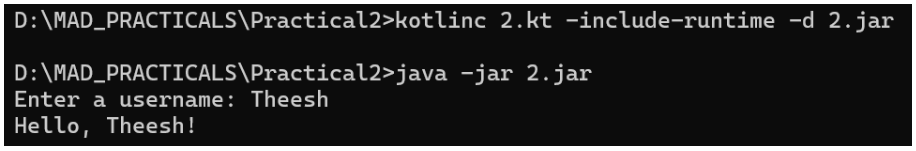
Index No : 22000461

1. Write a Kotlin program that checks whether a given number is even or odd.



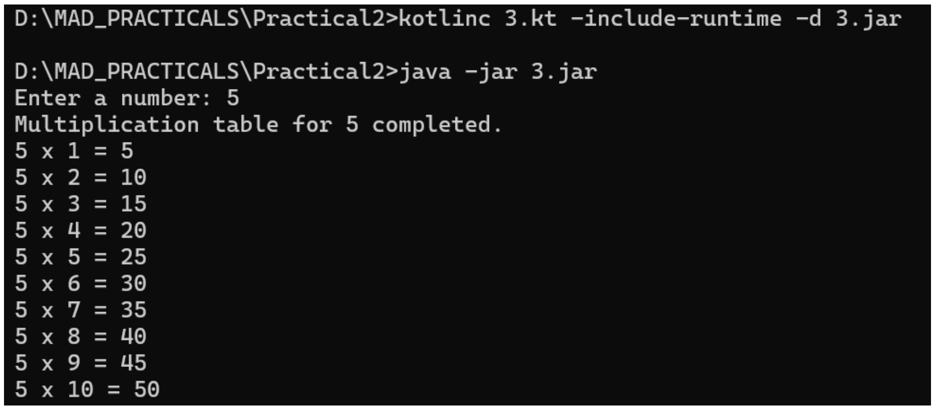
```
D:\MAD_PRACTICALS>java -jar Practical2.jar
Enter an integer: 2
2 is even
```

2. Create a program that takes a user's name and prints a greeting message using string interpolation.



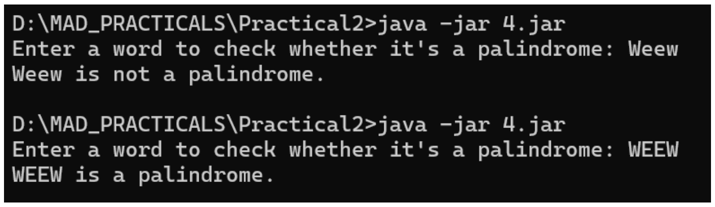
```
D:\MAD_PRACTICALS\Practical2>kotlinc 2.kt -include-runtime -d 2.jar

D:\MAD_PRACTICALS\Practical2>java -jar 2.jar
Enter a username: Theesh
Hello, Theesh!
```

3. Write a Kotlin program that prints the multiplication table of a given number.



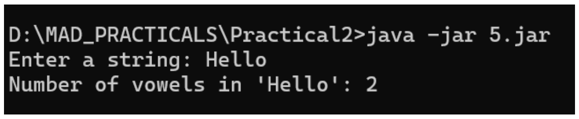
```
D:\MAD_PRACTICALS\Practical2>kotlinc 3.kt -include-runtime -d 3.jar

D:\MAD_PRACTICALS\Practical2>java -jar 3.jar
Enter a number: 5
Multiplication table for 5 completed.
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

4. Implement a Kotlin function that checks whether a string is a palindrome.



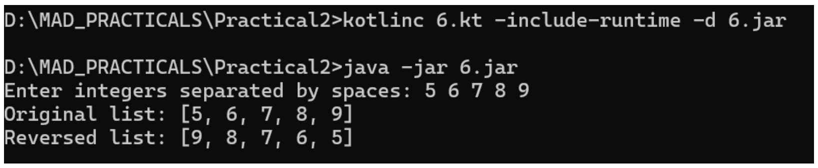
```
D:\MAD_PRACTICALS\Practical2>java -jar 4.jar
Enter a word to check whether it's a palindrome: Weew
Weew is not a palindrome.

D:\MAD_PRACTICALS\Practical2>java -jar 4.jar
Enter a word to check whether it's a palindrome: WEEW
WEEW is a palindrome.
```

5. Develop a program that counts the number of vowels in a given input string.

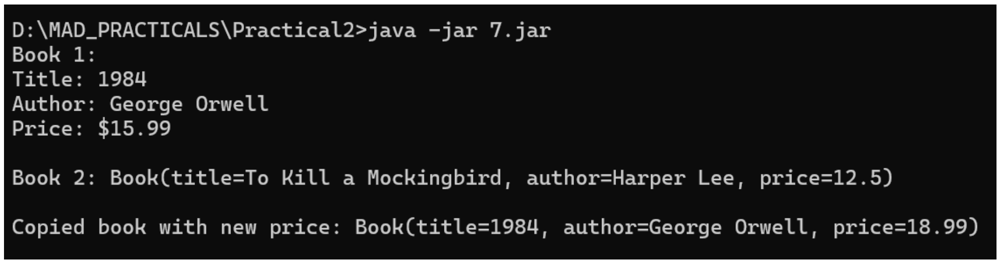


```
D:\MAD_PRACTICALS\Practical2>java -jar 5.jar
Enter a string: Hello
Number of vowels in 'Hello': 2
```

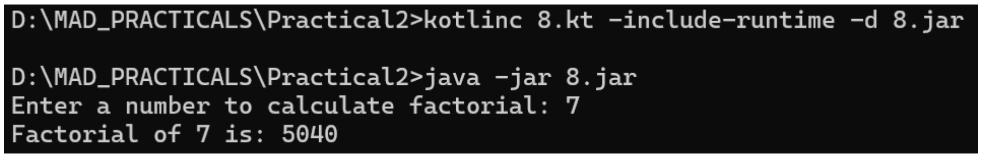
6. Write a Kotlin program that reverses a given list of integers.


```
D:\MAD_PRACTICALS\Practical2>kotlinc 6.kt -include-runtime -d 6.jar

D:\MAD_PRACTICALS\Practical2>java -jar 6.jar
Enter integers separated by spaces: 5 6 7 8 9
Original list: [5, 6, 7, 8, 9]
Reversed list: [9, 8, 7, 6, 5]
```

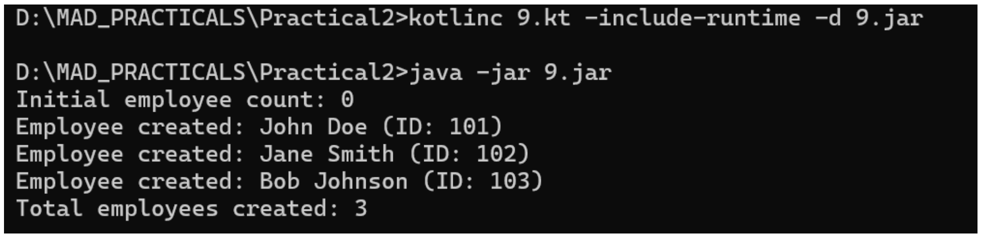
7. Create a data class Book with properties title, author, and price, and demonstrate how to create an instance and print its properties.


```
D:\MAD_PRACTICALS\Practical2>java -jar 7.jar
Book 1:
Title: 1984
Author: George Orwell
Price: $15.99

Book 2: Book(title=To Kill a Mockingbird, author=Harper Lee, price=12.5)

Copied book with new price: Book(title=1984, author=George Orwell, price=18.99)
```

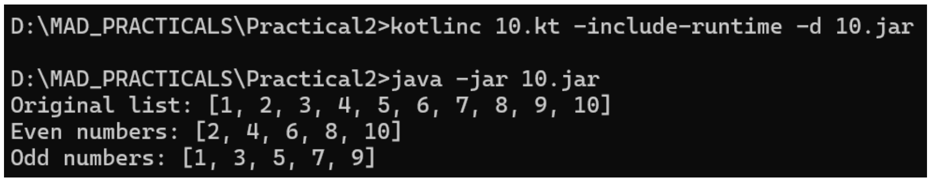
8. Develop a simple program that calculates the factorial of a given number using a while loop.


```
D:\MAD_PRACTICALS\Practical2>kotlinc 8.kt -include-runtime -d 8.jar

D:\MAD_PRACTICALS\Practical2>java -jar 8.jar
Enter a number to calculate factorial: 7
Factorial of 7 is: 5040
```

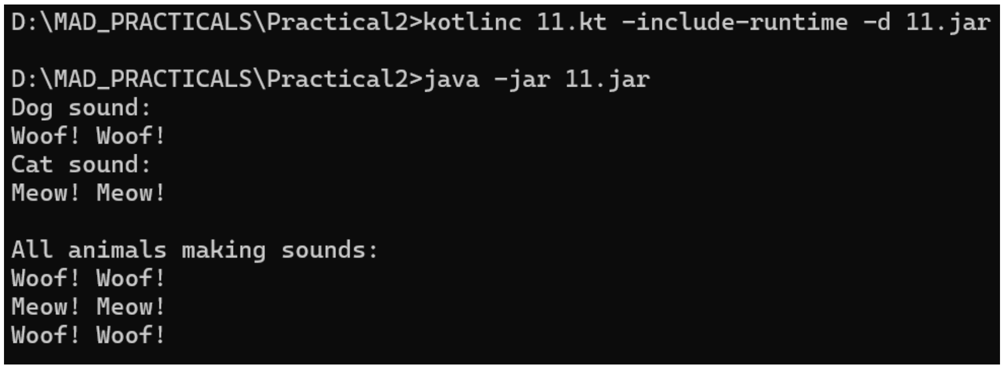
9. Create a class Employee with a companion object that keeps track of the number of employees created.


```
D:\MAD_PRACTICALS\Practical2>kotlinc 9.kt -include-runtime -d 9.jar

D:\MAD_PRACTICALS\Practical2>java -jar 9.jar
Initial employee count: 0
Employee created: John Doe (ID: 101)
Employee created: Jane Smith (ID: 102)
Employee created: Bob Johnson (ID: 103)
Total employees created: 3
```

10. Write a Kotlin function that filters out even numbers from a list using lambda expressions.

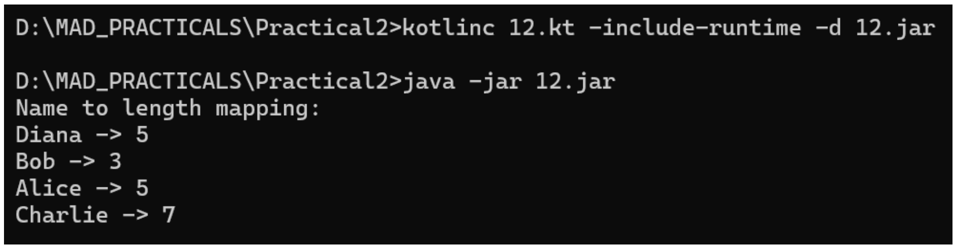

```
D:\MAD_PRACTICALS\Practical2>kotlinc 10.kt -include-runtime -d 10.jar

D:\MAD_PRACTICALS\Practical2>java -jar 10.jar
Original list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Even numbers: [2, 4, 6, 8, 10]
Odd numbers: [1, 3, 5, 7, 9]
```

11. Create an abstract class Animal with an abstract method makeSound(). Implement it in subclasses like Dog and Cat, and demonstrate their usage.



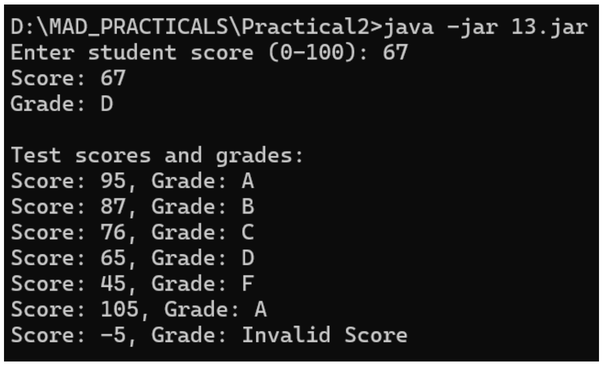
```
D:\MAD_PRACTICALS\Practical2>kotlinc 11.kt -include-runtime -d 11.jar

D:\MAD_PRACTICALS\Practical2>java -jar 11.jar
Dog sound:
Woof! Woof!
Cat sound:
Meow! Meow!

All animals making sounds:
Woof! Woof!
Meow! Meow!
Woof! Woof!
```

12. Write a Kotlin function that takes a list of names and returns a map where each name is mapped to its length.



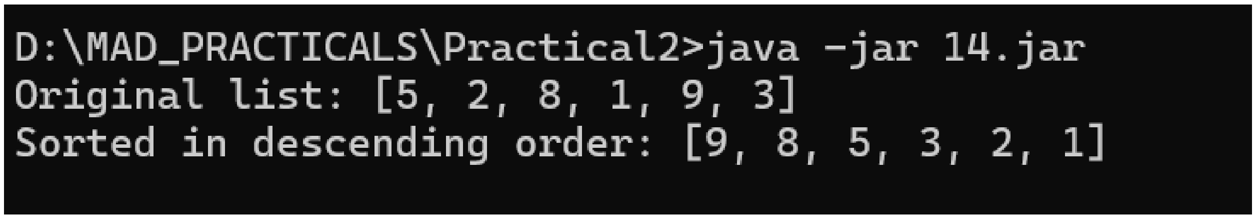
```
D:\MAD_PRACTICALS\Practical2>kotlinc 12.kt -include-runtime -d 12.jar

D:\MAD_PRACTICALS\Practical2>java -jar 12.jar
Name to length mapping:
Diana -> 5
Bob -> 3
Alice -> 5
Charlie -> 7
```

13. Use Kotlin's when expression to create a grading system that assigns grades based on the student's score (0–100).



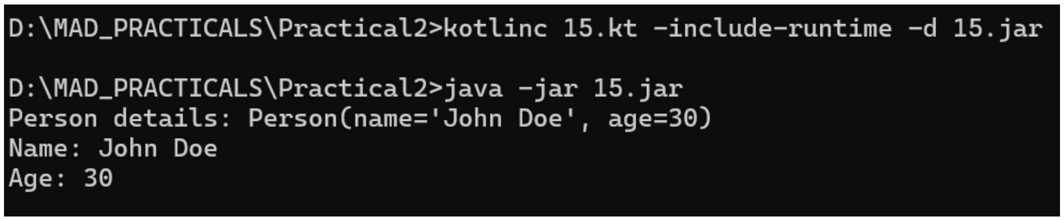
```
D:\MAD_PRACTICALS\Practical2>java -jar 13.jar
Enter student score (0-100): 67
Score: 67
Grade: D

Test scores and grades:
Score: 95, Grade: A
Score: 87, Grade: B
Score: 76, Grade: C
Score: 65, Grade: D
Score: 45, Grade: F
Score: 105, Grade: A
Score: -5, Grade: Invalid Score
```

14. Create a Kotlin program that sorts a list of integers in descending order using a custom comparator.



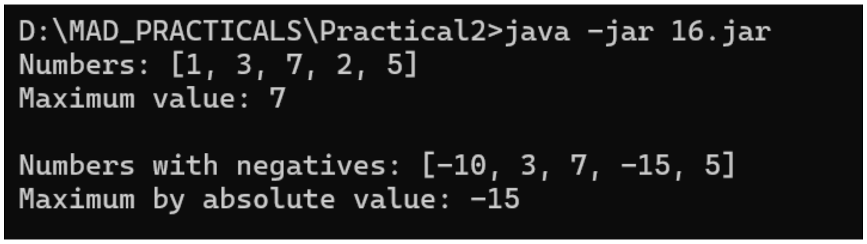
```
D:\MAD_PRACTICALS\Practical2>java -jar 14.jar
Original list: [5, 2, 8, 1, 9, 3]
Sorted in descending order: [9, 8, 5, 3, 2, 1]
```

15. Create a class Person with properties name and age. Use property delegation to delegate these properties to another class called PersonDelegate.



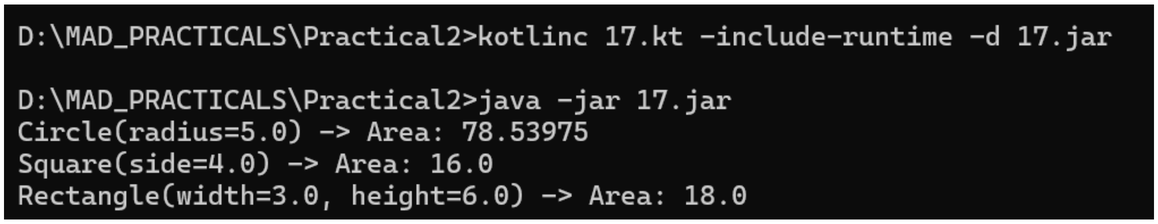
```
D:\MAD_PRACTICALS\Practical2>kotlinc 15.kt -include-runtime -d 15.jar

D:\MAD_PRACTICALS\Practical2>java -jar 15.jar
Person details: Person(name='John Doe', age=30)
Name: John Doe
Age: 30
```

16. Write a function findMax that takes a list of integers and a higher-order function to define the comparison logic. Use it to find the maximum in the list [1, 3, 7, 2, 5].



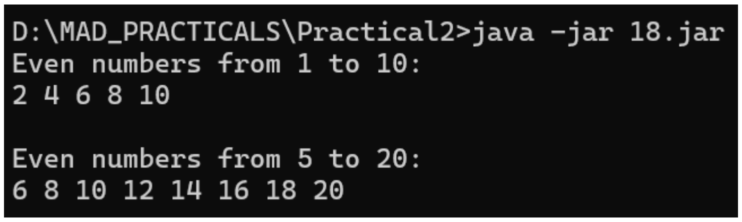
```
D:\MAD_PRACTICALS\Practical2>java -jar 16.jar
Numbers: [1, 3, 7, 2, 5]
Maximum value: 7

Numbers with negatives: [-10, 3, 7, -15, 5]
Maximum by absolute value: -15
```

17. Define a sealed class Shape with subclasses Circle, Square, and Rectangle. Write a function calculateArea() to compute the area based on the shape.



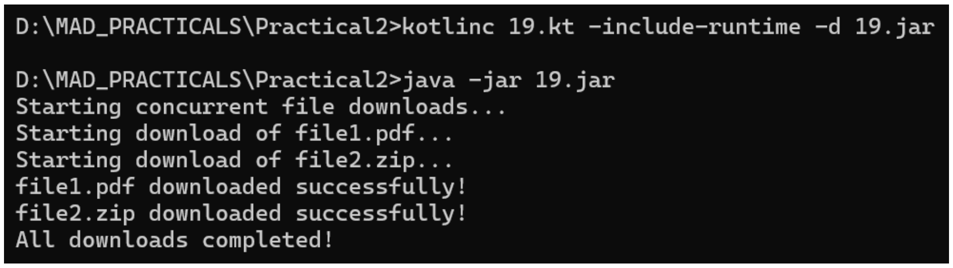
```
D:\MAD_PRACTICALS\Practical2>kotlinc 17.kt -include-runtime -d 17.jar

D:\MAD_PRACTICALS\Practical2>java -jar 17.jar
Circle(radius=5.0) -> Area: 78.53975
Square(side=4.0) -> Area: 16.0
Rectangle(width=3.0, height=6.0) -> Area: 18.0
```

18. Implement a custom iterator that iterates only over even numbers within a given range. Demonstrate it from 1 to 10.



```
D:\MAD_PRACTICALS\Practical2>java -jar 18.jar
Even numbers from 1 to 10:
2 4 6 8 10

Even numbers from 5 to 20:
6 8 10 12 14 16 18 20
```

19. Write a Kotlin program using runBlocking and launch to simulate downloading two files concurrently. Print a message after each download finishes.



```
D:\MAD_PRACTICALS\Practical2>kotlinc 19.kt -include-runtime -d 19.jar

D:\MAD_PRACTICALS\Practical2>java -jar 19.jar
Starting concurrent file downloads...
Starting download of file1.pdf...
Starting download of file2.zip...
file1.pdf downloaded successfully!
file2.zip downloaded successfully!
All downloads completed!
```