1. There is a colony of 8 cells arranged in a straight line where each day every cell competes with its adjacent cells (neighbour). Each day, for each cell, if its neighbours are both active and both inactive, the cell becomes inactive the next day, otherwise it becomes active the next day.

   Assumptions: The two cells on the ends have single adjacent cell, so the other adjacent cell can be assumed to be always inactive.
   Even after updating the cell state. Consider its previous state for updating the state of other cells. Update the cell information of all cells simultaneously.
   Write a function cell Compete which takes one 8 element array of integer's cells representing the current state of 8 cells and one integer days representing the number of days to simulate.
   An integer value of 1 represents an active cell and value of 0 represents an inactive cell. **(8cell.cpp)**

2. To check is the number is abundant or not. An abundant number is a number for which the sum of its proper divisors is greater than the number itself. **(abundant.cpp)**
3. To check is the number is armstrong or not. A number is an Armstrong number when the sum of nth power of each digit is equal to the number itself.**(armstrong.cpp)**
4. To check is the number is strong or not. A strong number is a number in which the sum of the factorial of the digits is equal to the number itself.**(strong.cpp)**
5. To convert binary to decimal when binary is given as a string.**(b2dstring.cpp)**
6. To convert binary to decimal when binary is given as a number.**(binary2decimal.cpp)**
7. To convert decimal to binary when binary is given as a string.**(decimal2binary.cpp)**
8. To find the factorial of a number.**(factorial.cpp)**
9. To find the GCD and LCM of two digits.**(gcd_lcm.cpp)**
10. To print the Fibonacci series upto n. **(fibonacci.cpp)**
11. To print all the digits in the given string.**(isdigit.cpp)**
12. To convert the given string into lowercase.**(islower.cpp)**
13. To convert the given string into uppercase.**(isupper.cpp)**
14. To print a pattern of hollow square using * .**(hollowsquare.cpp)**
15. To print a pattern of square using *.**(square.cpp)**
16. To print the all special symbols like ' !@#$%' in the given string.**(ispunct.cpp)**
17. To rotate the given array left by n times.**(leftrotate.cpp)**
18. To rotate the given array right by n times.**(rightrotate.cpp)**
19. To find the maximum from the array using stl.**(max.cpp)**

20. To find the minimum from the array using stl.**(min.cpp)**
21. To check whether a number is prime or not.**(prime.cpp)**
22. To print all the prime numbers upto n.**(print_n_prime.cpp)**
23. To print a pyramid using numbers.**(pyramid-1.cpp)**
24. To print a pyramid using alphabets.**(pyramidABC.cpp)**
25. To reverse, sort(increasing and decreasing order) using inbuilt algorithms libraries in C++.**(reverse.cpp)**
26. To reverse a given number. (You can use this function to check whether a number is palindrome or not by just storing the number before modifying and after reversing you can check if (oldnum == reverse_num) then true otherwise false).**(reverse_num.cpp)**
27. To remove the spaces between the given string. **(string.cpp)**
28. To convert a given string in uppercase using stl libraries.**(toupper.cpp)**
29. To convert a given string in lowercase using stl libraries.**(tolower.cpp)**