Objective:

- Focus the use of const keyword with data member and functions.
- Targets the use/purpose of class level information.

Challenge: ***Token/Counter machine***

We need to develop a software for Automatic Token generation machine.

Following features should be supported by the machine:

1. The token generated by the machine should have a unique token-id.
2. Expected wait time for the particular token number to be serviced.
3. The company, who use/install the machine must be able to define the average processing time for a client and on the basis of this given information, machine should be able to calculate/print the expected waiting time for particular token number keeping in view the active tokens (tokens, which needs to be serviced yet.)
4. Machine should be able to generate token numbers that will start from 1 and may go up to 4294967295 (max value in an unsigned int).
5. Company (machine operator) should also be able to set the limit on the count of active tokens. The restriction on the count of active tokens will help company to manage the persons to be serviced and will also be helpful to maintain the COVID SOPs.
6. There will be an option to reset the machine which means the machine will consider all tokens expired and will start generating the tokens again from token-id # 1.

Considering the requirements/features listed above, You job is to implement the class named 'TokenMachine'. See the explanation and sample run below to understand the behaviour.

```
class TokenMachine
{
    static unsigned int _tokenNumber;
         //Represents the next token number that will be allotted to a customer.

    static const unsigned int _MAX_LIVE_TOKENS;
         //Represents the maximum live token a system may have. Very useful considering COVID SOPs

    static const double _AVERAGE_PROCESSING_TIME_IN_MINUTES;
         //As the name suggests, average processing time to service a token/customer.

    static unsigned int _activeTokens;
         //count of active tokens that needs to be serviced yet.

      //Used to initialize _MAX_LIVE_TOKENS. Don't need to change the coding of this function. We
      assume, the default live tokens a machine may produce is 5 but that can be either by passing
      argument to this function or by taking console input from user. Observe the way it is called at
      global level to initialize _MAX_LIVE_TOKENS.
   static int setMaxLiveTokens(int num=5)
   {
       //we may write code to take input from keyboard and return that value.
       //cin>>num
       //but here we are using a default setting of max Live token i.e. 5
       return num;
   }

     //Same strategy as we used for setMaxLiveTokens
   static double setAverageProcessingTime(double tm=3)
   {
       //we may write code to take input from keyboard and return that value.
       //cin>>tm
       //but here we are using a default setting of average processing time i.e. 3
```

```
        return tm;
    }

public:
    static String getNextToken();
        //It is called when next token is to be generated from the machine. It returns a receipt
        (technically a String object) which has detail about token number and other related
        information. See sample run.

    static unsigned int getActiveTokensCount();
        //Return the count of active tokens.

    static void personIsServiced();
        //It is called when a token/person is serviced. Decide/Think yourself that which class level
        information, it should update.

    static unsigned int getCountOfPersonsServiced();
        //It returns the count of persons serviced since last machine reset.

    static void resetMachine()
    {
        _tokenNumber = 0;
        _activeTokens = 0;
    }
};

unsigned int TokenMachine::_tokenNumber = 0;
const unsigned int TokenMachine::_MAX_LIVE_TOKENS = TokenMachine::setMaxLiveTokens();
const double TokenMachine::_AVERAGE_PROCESSING_TIME_IN_MINUTES =
TokenMachine::setAverageProcessingTime();
unsigned int TokenMachine::_activeTokens = 0;
```

Sample Run

int main() {	
TokenMachine::getNextToken().display(); cout<<"\n\n";	Token Id: 1 Persons before you in line are: 0 Expected Waiting Time: 0 minutes
TokenMachine::personIsServiced();	
TokenMachine::getNextToken().display(); cout<<"\n\n";	Token Id: 2 Persons before you in line are: 0 Expected Waiting Time: 0 minutes
TokenMachine::getNextToken().display(); cout<<"\n\n";	Token Id: 3 Persons before you in line are: 1 Expected Waiting Time: 3 minutes
TokenMachine::getNextToken().display(); cout<<"\n\n";	Token Id: 4 Persons before you in line are: 2 Expected Waiting Time: 6 minutes
TokenMachine::personIsServiced();	
TokenMachine::getNextToken().display(); cout<<"\n\n";	Token Id: 5 Persons before you in line are: 2 Expected Waiting Time: 6 minutes
TokenMachine::getNextToken().display(); cout<<"\n\n";	Token Id: 6 Persons before you in line are: 3 Expected Waiting Time: 9 minutes
TokenMachine::getNextToken().display(); cout<<"\n\n";	Token Id: 7 Persons before you in line are: 4 Expected Waiting Time: 12 minutes
TokenMachine::getNextToken().display(); cout<<"\n\n";	We are Sorry. There can't be more than 5 tokens in system for service. Please wait for some clients to get serviced
TokenMachine::getNextToken().display(); cout<<"\n\n";	We are Sorry. There can't be more than 5 tokens in system for service. Please wait for some clients to get serviced
TokenMachine::getNextToken().display(); cout<<"\n\n";	We are Sorry. There can't be more than 5 tokens in system for service. Please wait for some clients to get serviced
TokenMachine::personIsServiced();	
cout<<"Count of Persons Serviced: " <<	Count of Persons Serviced: 3

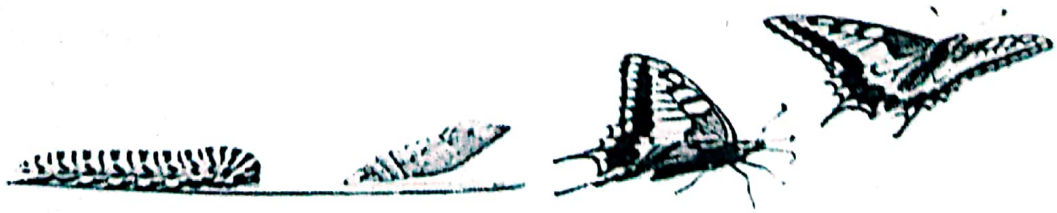
`TokenMachine::getCountOfPersonsServiced()<<"\n\n";`	
`TokenMachine::personIsServiced();`	

`TokenMachine::getNextToken().display(); cout<<"\n\n";`	`Token Id: 8`
	`Persons before you in line are: 3`
	`Expected Waiting Time: 9 minutes`

`TokenMachine::getNextToken().display(); cout<<"\n\n";`	`Token Id: 9`
	`Persons before you in line are: 4`
	`Expected Waiting Time: 12 minutes`

`cout<<endl;`
`return 0;`
`}`	

Note: When it is not possible to generate next token because of having live tokens equal to _MAX_LIVE_TOKENS, then getNextToken returns a message to wait for some person to get serviced, you are not supposed to hard code the _MAX_LIVE_TOKENS information in it. It means if we change the value of _MAX_LIVE_TOKENS then this "wait message" must not need any change at all.



"Just when the caterpillar thought the world was ending.