# EXP 7 STUDY CODE OPTIMIZATION

**CODE OPTIMIZATION:** The process of code optimization involves

* Eliminating the unwanted code lines
* Rearranging the statements of the code

## CODE OPTIMIZATION TECHNIQUES:

1. **Compile Time Evaluation**

Two techniques that falls under compile time evaluation are-

## Constant Folding

In this technique,

* + As the name suggests, it involves folding the constants.
  + The expressions that contain the operands having constant values at compile time are evaluated.
  + Those expressions are then replaced with their respective results.

## Constant Propagation

In this technique,

* + If some variable has been assigned some constant value, then it replaces that variable with its constant value in the further program during compilation.
  + The condition is that the value of variable must not get alter in between.

## Example:

pi = 3.14

radius = 10

Area of circle = pi x radius x radius Here,

* + This technique substitutes the value of variables ‘pi’ and ‘radius’ at compile time.
  + It then evaluates the expression 3.14 x 10 x 10.
  + The expression is then replaced with its result 314.
  + This saves the time at run time.

## Common Sub-Expression Elimination

The expression that has been already computed before and appears again in the code for computation is called as **Common Sub-Expression**.

In this technique,

* + As the name suggests, it involves eliminating the common sub expressions.
  + The redundant expressions are eliminated to avoid their re-computation.
  + The already computed result is used in the further program when required.

## Example

1. **Code Movement**

In this technique,

* + As the name suggests, it involves movement of the code.
  + The code present inside the loop is moved out if it does not matter whether it is present inside or outside.
  + Such a code unnecessarily gets execute again and again with each iteration of the loop.
  + This leads to the wastage of time at run time.

## Example:

|  |  |
| --- | --- |
| **Code Before Optimization** | **Code After Optimization** |
| for ( int j = 0 ; j < n ; j ++)  {  x = y + z ; a[j] = 6 x j  } | x = y + z ;  for ( int j = 0 ; j < n ; j ++)  {  a[j] = 6 x j;  } |

1. **Dead Code Elimination**

In this technique,

* + As the name suggests, it involves eliminating the dead code.
  + The statements of the code which either never executes or are unreachable or their output is never used are eliminated.

## Example:

**5. Strength Reduction**

In this technique,

* + As the name suggests, it involves reducing the strength of expressions.
  + This technique replaces the expensive and costly operators with the simple and cheaper ones.