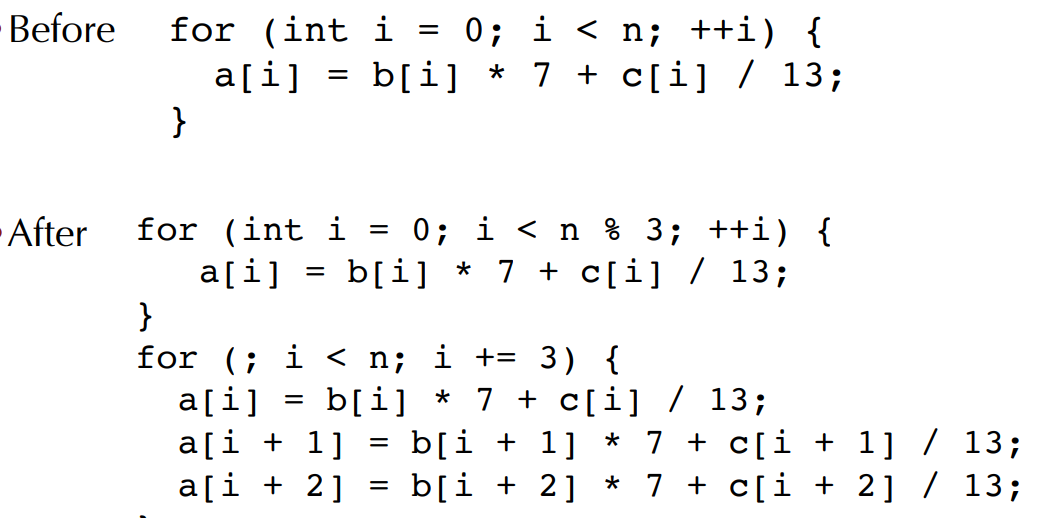
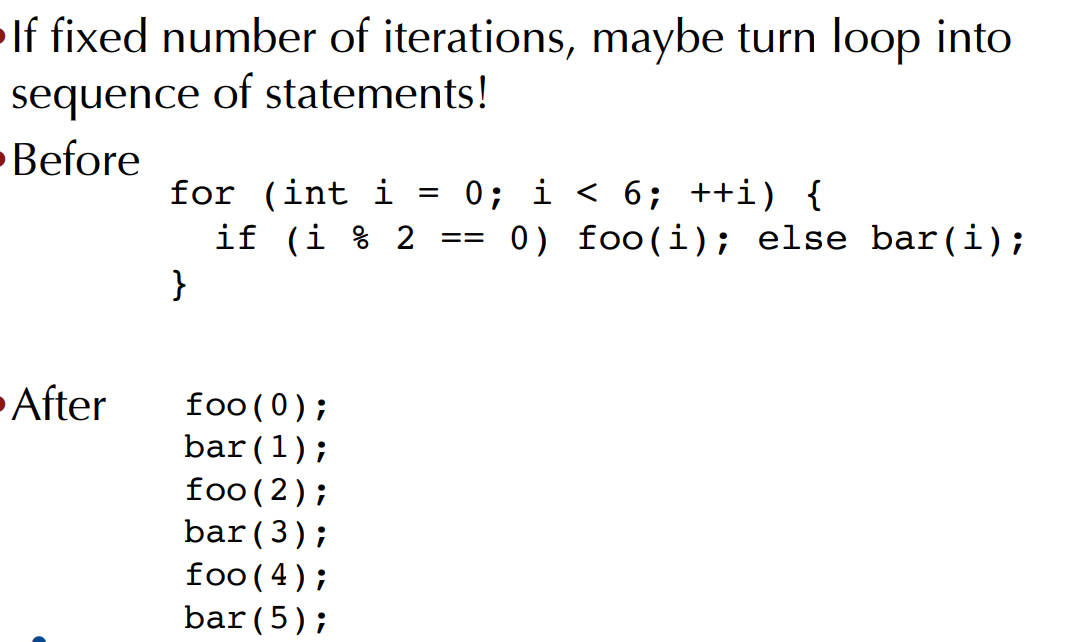
**Experiment no – 09**

**Aim: Write a program to demonstrate loop unrolling and loop splitting for the given code sequence containing loop.**

**Theory: -**

Loop unrolling is a loop transformation technique that helps to optimize the execution time of a program. We basically remove or reduce iterations. Loop unrolling increases the program’s speed by eliminating loop control instruction and loop test instructions.





Example:

**// This program does not uses loop unrolling.**

#include<stdio.h>

int main(void)

{

for (int i=0; i<5; i++)

printf("Hello\n"); //print hello 5 times

return 0;

}

// **This program uses loop unrolling.**

#include<stdio.h>

int main(void)

{

// unrolled the for loop in program 1

printf("Hello\n");

printf("Hello\n");

printf("Hello\n");

printf("Hello\n");

printf("Hello\n");

return 0;

}

**Loop splitting**(or loop peeling) is a compiler optimization technique. It attempts to simplify a loop or eliminate dependencies by breaking it into multiple loops which have the same bodies but iterate over different contiguous portions of the index range.

A useful special case is loop peeling, which can simplify a loop with a problematic first (or first few) iteration by performing that iteration separately before entering the loop.

Here is an example of loop peeling. Suppose the original code looks like this:

**p = 10; for (i=0; i<10; ++i) { y [i] = x [i] + x [p] ; p = i; }**

In the above code, only in the 1st iteration is p=10. For all other iterations p=i-1. We get the following after loop peeling:

**y [0] = x [0] + x [10] ; for (i=1; i<10; ++i) { y [i] = x [i] + x [i-1] ; }**