Experiment no – 10

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.

Before
```
for (int i = 0; i < n; ++i) {
  a[i] = b[i] * 7 + c[i] / 13;
}
```

After
```
for (int i = 0; i < n % 3; ++i) {
   a[i] = b[i] * 7 + c[i] / 13;
}
for (; i < n; i += 3) {
  a[i] = b[i] * 7 + c[i] / 13;
  a[i + 1] = b[i + 1] * 7 + c[i + 1] / 13;
  a[i + 2] = b[i + 2] * 7 + c[i + 2] / 13;
```

If fixed number of iterations, maybe turn loop into sequence of statements!

Before
```
for (int i = 0; i < 6; ++i) {
  if (i % 2 == 0) foo(i); else bar(i);
}
```

After
```
foo(0);
bar(1);
foo(2);
bar(3);
foo(4);
bar(5);
```

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 bodi es 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 fi rst 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] ; }