1. Convert this program to SSA form:

```
1. \( m \leftarrow 0 \)
2. \( v \leftarrow 0 \)
3. if \( v < n \) then \( x \leftarrow M[r] \)
   false \( s \leftarrow s + x \)
   if \( s \leq m \) then \( m \leftarrow s \)
4. return \( m \)
5. if \( r \geq n \) then \( r \leftarrow r + 1 \)
6. \( v \leftarrow v + 1 \)
7. \( r \leftarrow v \)
8. \( s \leftarrow 0 \)
```

Show your work after each stage:

a) Add a start node containing initializations of all variables.

b) Draw the dominator tree.

c) Calculate dominance frontiers.

d) Insert \( \phi \)-functions.

e) Add subscripts to variables.

f) Convert back from SSA form by inserting move instructions in place of \( \phi \)-functions.
Exercise 12.1.1: In Fig. 12.10 is a C program with two function pointers, \( p \) and \( q \). \( N \) is a constant that could be less than or greater than 10. Note that the program results in an infinite sequence of calls, but that is of no concern for the purposes of this problem.

a) Identify all the call sites in this program.

b) For each call site, what can \( p \) point to? What can \( q \) point to?

c) Draw the call graph for this program.

d) Describe all the call strings for \( f \) and \( g \).

```c
int (*p)(int);
int (*q)(int);

int f(int i) {
    if (i < 10)
        {p = &g; return (*q)(i);}  
    else
        {p = &f; return (*p)(i);}  
}

int g(int j) {
    if (j < 10)
        {q = &f; return (*p)(j);}  
    else
        {q = &g; return (*q)(j);}  
}

void main() {
    p = &f;
    q = &g;
    (*p)((*q)(N));
}
```

Figure 12.10: Program for Exercise 12.1.1