```
int main(){
    int z;
    z = foo(10,20);
    return z;
}
int foo(int a, int b)
{
    int c,temp3;
    int temp1,temp2;

    c = a*b;
    temp1 = a+b;
    temp2 = a/b;
    if(a>40){
        temp3 = temp1 + temp2;
    }
    else {
        temp3 = temp1 - temp2;
    }
    c = c + temp3;
    return c;
}
```

(a) Original C Code

```
char *main()
{
    int llvm_tmp3;
    llvm_tmp_3 = rewritten_foo(10,20);
    return llvm_tmp3;
}

int rewritten_foo(int llvm_Arg1,
                  int llvm_Arg2)
{
    int llvm_tmp4;
    int llvm_tmp1 = llvm_Arg1 * llvm_Arg2;
    int llvm_tmp2 = llvm_Arg1 + llvm_Arg2;
    int llvm_tmp3 = llvm_Arg1 - llvm_Arg2;
    if (llvm_Arg1 <= 40){
      llvm_tmp4 =  llvm_tmp2 + llvm_tmp3;
    }
    else {
      llvm_tmp4 = llvm_tmp2 - llvm_tmp3;
    }
    llvm_tmp4 = llvm_tmp1 + llvm_tmp4
    return llvm_tmp4;
}
```

(d) Recovered C Code with abstract stack and symbol promotion

```
//Global Stack Pointer
int* llvm_ESP;

char *main()
{
    llvm_ESP = llvm_ESP-2; //Local Allocation

    llvm_ESP2[1] = 20; //Outgoing argument
    llvm_ESP2[0] = 10;
    int llvm_tmp_3 = rewritten_foo();
    return llvm_tmp3;
}

int rewritten_foo()
{
    int* llvm_EBP = llvm_ESP;
              //Local Frame Pointer
    llvm_ESP = llvm_ESP-10;
              //Local Allocation

    int tmpIn1 = llvm_EBP[0]; //Incoming Arg
    int tmpIn2; = llvm_EBP[1];

    int  llvm_tmp1 = tmpIn1*tmpIn2;
    llvm_ESP[1] = llvm_tmp1;

    int  llvm_tmp2 = tmpIn1+tmpIn2;
    llvm_ESP[2] = llvm_tmp2;

    int llvm_tmp3 = tmpIn1-tmpIn2;
    llvm_ESP[3] = llvm_tmp3;

    int llvm_tmpIn3 = llvm_EBP[0];
    if (llvm_tmpIn3 > 40){
      int llvm_tmp5 = llvm_ESP[2];
      int llvm_tmp6 = llvm_ESP[3];
      llvm_ESP[5] = llvm_tmp5 + llvm_tmp6;
    }
    else {
      int llvm_tmp7 = llvm_ESP[2];
      int llvm_tmp8 = llvm_ESP[3];
      llvm_ESP[5] = llvm_tmp7 - llvm_tmp8;
    }
    int llvm_tmp9 = llvm_ESP[1];
    int llvm_tmp10 = llvm_ESP[5];
    int llvm_tmp11 = llvm_tmp9 +
              llvm_tmp10;
   return llvm_tmp11;
}
```

(b) Recovered C Code with physical stack

```
char *main()
{
   int llvm_ESP2[10];

   llvm_ESP2[1] = 20;
   llvm_ESP2[2] = 10;
   int llvm_tmp1 = llvm_ESP2[1];
   int llvm_tmp2 = llvm_ESP2[2];
   int llvm_tmp_3 =
        rewritten_foo(llvm_tmp2,
                      llvm_tmp1);
   return llvm_tmp3;
}

int rewritten_foo( int llvmArg1,
                   int llvm_Arg2)
{
    int llvm_ESP1[10];

    int llvm_tmp1 = llvm_Arg1*llvm_Arg2;
    llvm_ESP1[1] = llvm_tmp1;
    int llvm_tmp2 = llvm_Arg1+llvm_Arg2;
    llvm_ESP1[2] = llvm_tmp2;
    int llvm_tmp3 = llvm_Arg1 - llvm_Arg2;
    llvm_ESP1[3] = llvm_tmp3;

    if (llvm_Arg1 > 40) {
       int llvm_tmp5 = llvm_ESP1[2];
       int llvm_tmp6 = llvm_ESP1[3];
       llvm_ESP1[5] =
              llvm_tmp5 + llvm_tmp6;
    }
    else {
        int llvm_tmp7 = llvm_ESP1[2];
        int llvm_tmp8 = llvm_ESP1[3];
        llvm_ESP1[5] =
              llvm_tmp7 - llvm_tmp8;
    }

    int llvm_tmp9 = llvm_ESP1[1];
    int llvm_tmp10 = llvm_ESP1[5];
    int llvm_tmp11 = llvm_tmp9 +
                     llvm_tmp10;
    return llvm_tmp11;
}
```

(c) Recovered C Code with abstract stack