COMPILERS LAB PROJECT

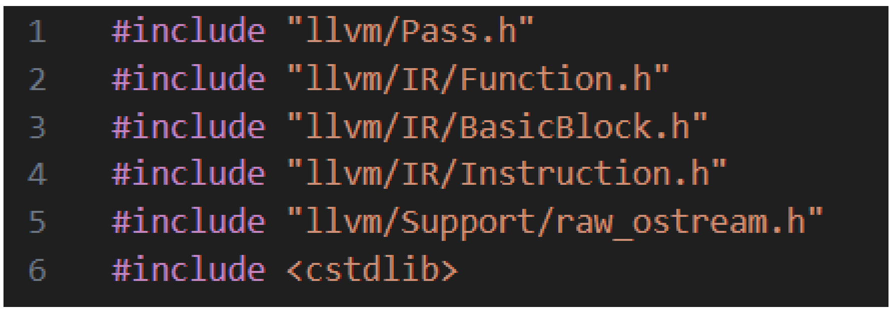
Team Members	Roll Numbers
Ritesh Kumar Gupta	2101171
Piyush Pranav	2101145
Piyush Patale	2101141

Problem Statement:

Design an LLVM pass to take a C++ application as input and find memory requirements for each basic block of the program. Also, print the ID of the desired basic block. You can use numbers to represent the basic block ID

Code Explanation:

Header Files



```
#include "llvm/Pass.h"
#include "llvm/IR/Function.h"
#include "llvm/IR/BasicBlock.h"
#include "llvm/IR/Instruction.h"
#include "llvm/Support/raw_ostream.h"
#include <cstdlib>
```

llvm/Pass.h - Passes in LLVM are transformations or analyses performed on LLVM intermediate representation (IR) code.

llvm/IR/Function.h - Functions in LLVM IR represent individual functions in the source program being compiled.

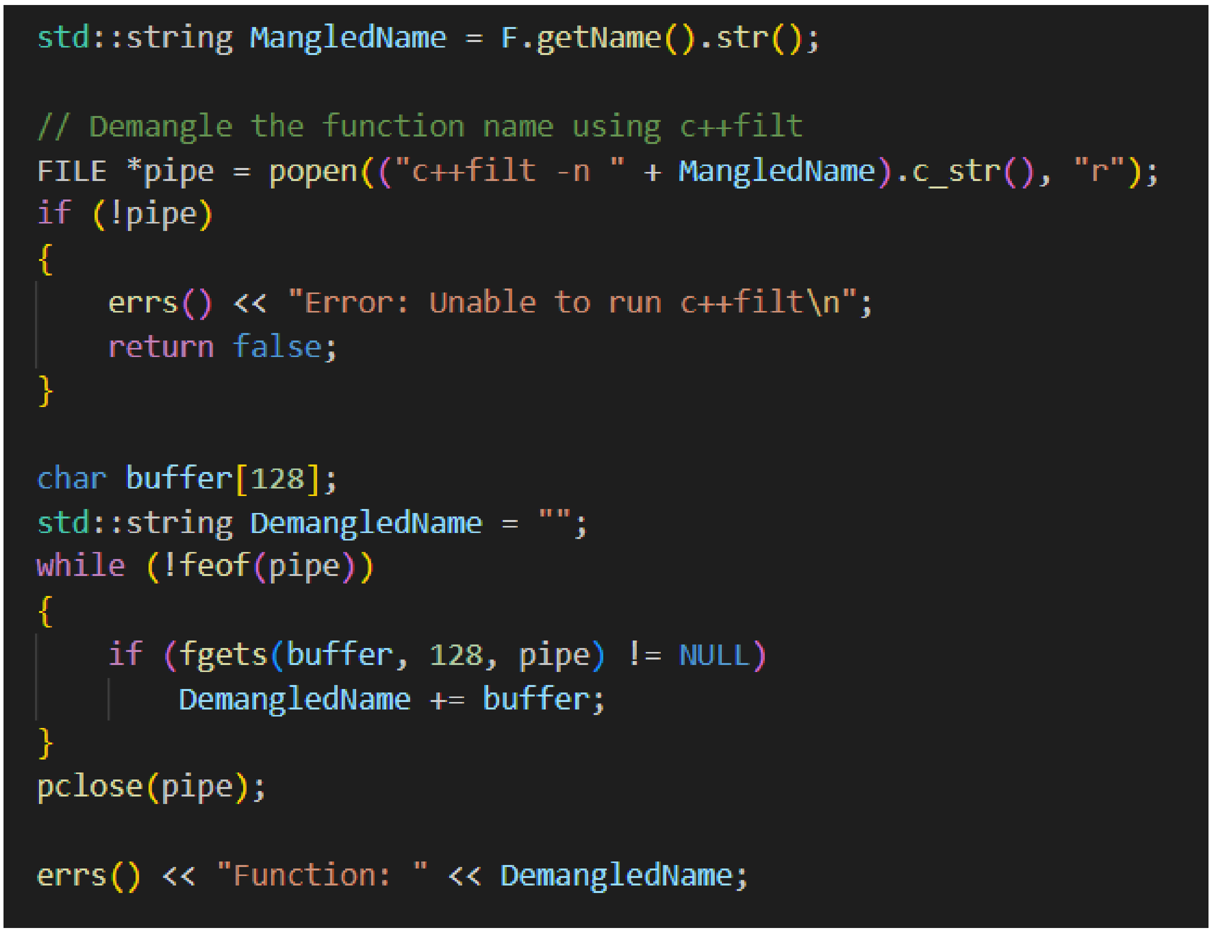
llvm/IR/BasicBlock.h - Basic blocks in LLVM IR represent a sequence of instructions with no branches except at the end.

llvm/IR/Instruction.h - Instructions in LLVM IR represent individual operations or statements within a basic block.

llvm/Support/raw_ostream.h - raw_ostream provides an abstraction for writing textual output to a stream, often used for printing debug information or analysis results.

<cstdlib> - cstdlib header provides general utilities, including memory allocation and conversion functions.

Demangling Function Name



```
std::string MangledName = F.getName().str();

// Demangle the function name using c++filt
FILE *pipe = popen(("c++filt -n " + MangledName).c_str(), "r");
if (!pipe)
{
    errs() << "Error: Unable to run c++filt\n";
    return false;
}

char buffer[128];
std::string DemangledName = "";
while (!feof(pipe))
{
    if (fgets(buffer, 128, pipe) != NULL)
        DemangledName += buffer;
}
pclose(pipe);

errs() << "Function: " << DemangledName;
```

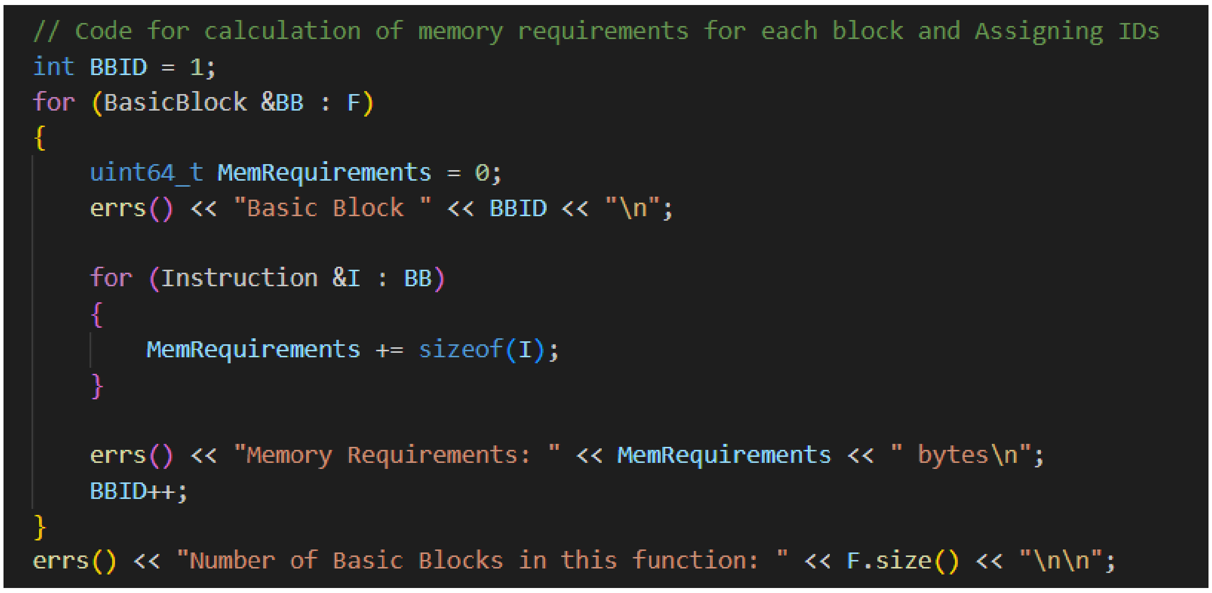
This part of the code aims to retrieve the mangled name of a C++ function represented in LLVM IR, demangle it using the c++filt utility, and print the demangled name.

The mangled name is first extracted from the LLVM IR, then passed as an argument to c++filt via a pipe to obtain the human-readable form of the function name.

- Mangled Name - _Z3fooii
- Demangled Name - void foo(int a, int b)

The demangled name is read from the output of c++filt and is printed to provide a more understandable representation of the function name.

Calculation Memory Requirements



```
// Code for calculation of memory requirements for each block and Assigning IDs
int BBID = 1;
for (BasicBlock &BB : F)
{
    uint64_t MemRequirements = 0;
    errs() << "Basic Block " << BBID << "\n";

    for (Instruction &I : BB)
    {
        MemRequirements += sizeof(I);
    }

    errs() << "Memory Requirements: " << MemRequirements << " bytes\n";
    BBID++;
}
errs() << "Number of Basic Blocks in this function: " << F.size() << "\n\n";
```

“for (BasicBlock &BB : F)” - Iterates over each basic block (BB) in the function (F). The loop uses a range-based for loop, iterating over each basic block within the function.

“for (Instruction &I : BB)” - Iterates over each instruction (I) within the current basic block (BB). This nested loop traverses through each instruction within the basic block.

“MemRequirements += sizeof(I)” - Calculates the memory requirements by adding the size of each instruction (I) to the MemRequirements variable. The size of each instruction is determined using the sizeof() operator, which returns the size in bytes.

“errs() << "Memory Requirements: " << MemRequirements << " bytes\n"” - Prints the calculated total memory requirements of instructions within the current basic block to the error stream.

Assigning IDs to the Basic Blocks

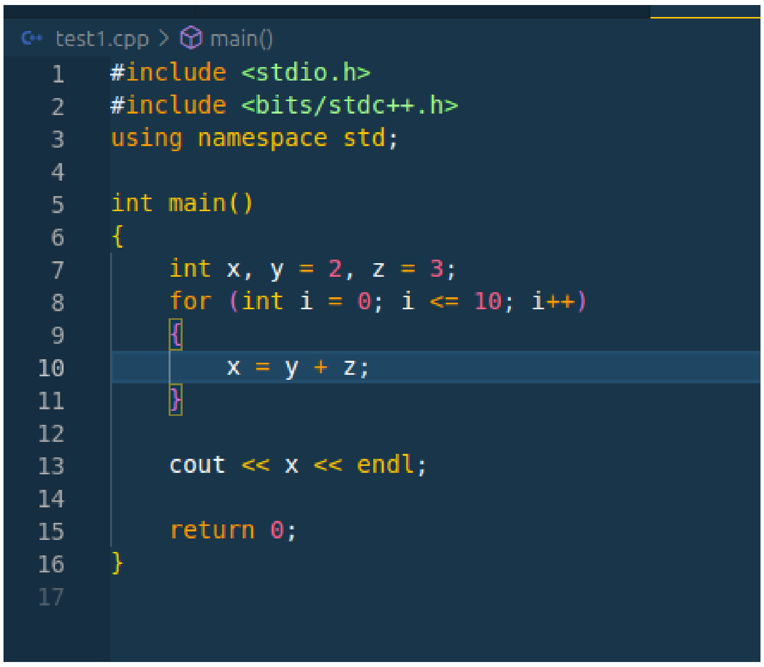
“int BBID = 1” - Initializes a variable BBID to keep track of the basic block identifier. It starts from 1.

“errs() << "Basic Block " << BBID << "\n"” and “BBID++” - Prints the ID of the current block and increments the variable for the next Basic Block.

Example Inputs and Outputs:

1. For Loop

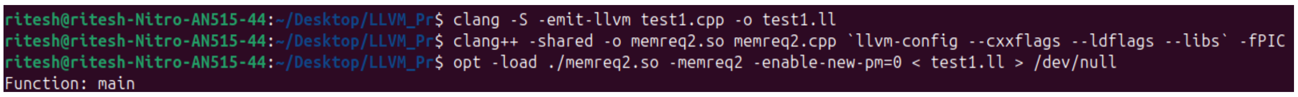
Code



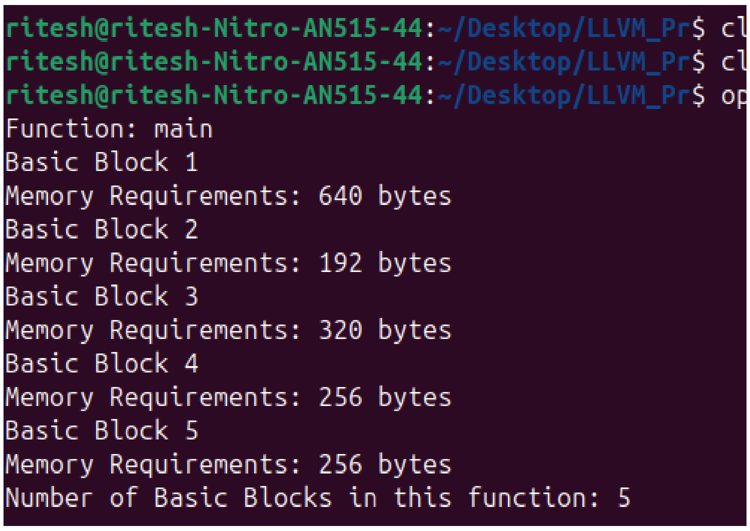
test1.cpp > main()

```cpp
#include <stdio.h>
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int x, y = 2, z = 3;
    for (int i = 0; i <= 10; i++)
    {
        x = y + z;
    }

    cout << x << endl;

    return 0;
}

```

Output



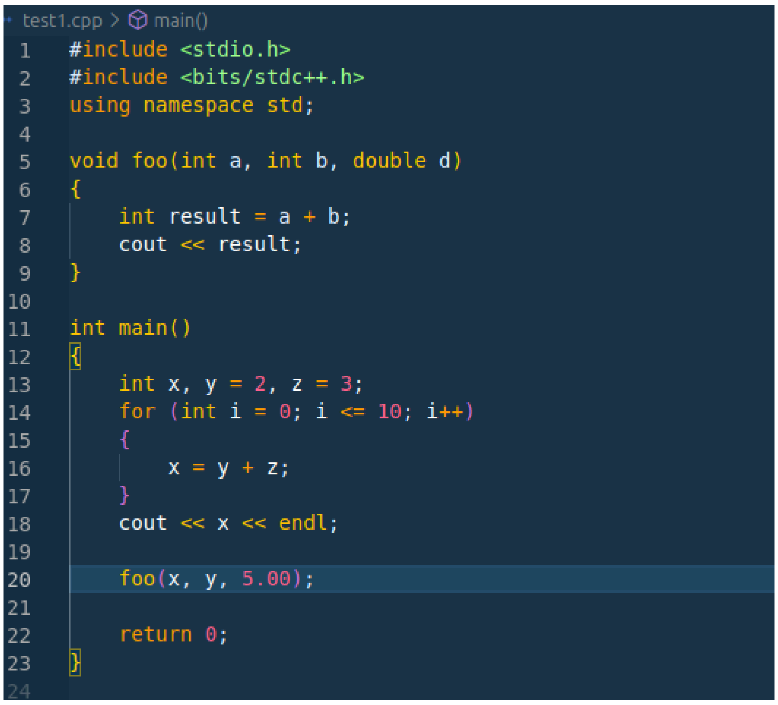
```
ritesh@ritesh-Nitro-AN515-44:~/Desktop/LLVM_Pr$ clang -S -emit-llvm test1.cpp -o test1.ll
ritesh@ritesh-Nitro-AN515-44:~/Desktop/LLVM_Pr$ clang++ -shared -o memreq2.so memreq2.cpp `llvm-config --cxxflags --ldflags --libs` -fPIC
ritesh@ritesh-Nitro-AN515-44:~/Desktop/LLVM_Pr$ opt -load ./memreq2.so -memreq2 -enable-new-pm=0 < test1.ll > /dev/null
Function: main
```



```
ritesh@ritesh-Nitro-AN515-44:~/Desktop/LLVM_Pr$ cl
ritesh@ritesh-Nitro-AN515-44:~/Desktop/LLVM_Pr$ cl
ritesh@ritesh-Nitro-AN515-44:~/Desktop/LLVM_Pr$ op
Function: main
Basic Block 1
Memory Requirements: 640 bytes
Basic Block 2
Memory Requirements: 192 bytes
Basic Block 3
Memory Requirements: 320 bytes
Basic Block 4
Memory Requirements: 256 bytes
Basic Block 5
Memory Requirements: 256 bytes
Number of Basic Blocks in this function: 5
```

2. For Loop + Another Function

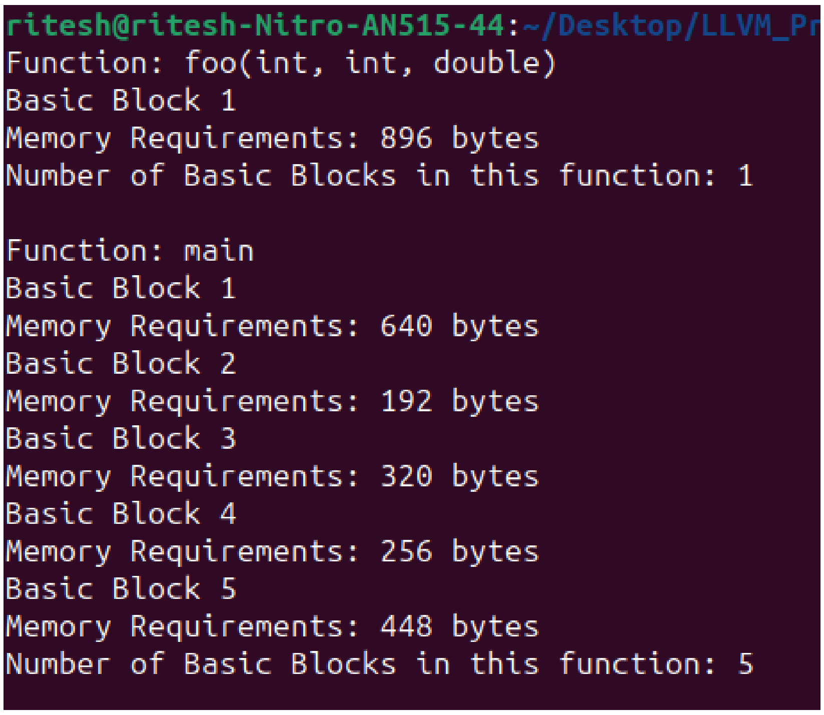
Code



test1.cpp > main()

```cpp
#include <stdio.h>
#include <bits/stdc++.h>
using namespace std;

void foo(int a, int b, double d)
{
    int result = a + b;
    cout << result;
}

int main()
{
    int x, y = 2, z = 3;
    for (int i = 0; i <= 10; i++)
    {
        x = y + z;
    }
    cout << x << endl;

    foo(x, y, 5.00);

    return 0;
}
```

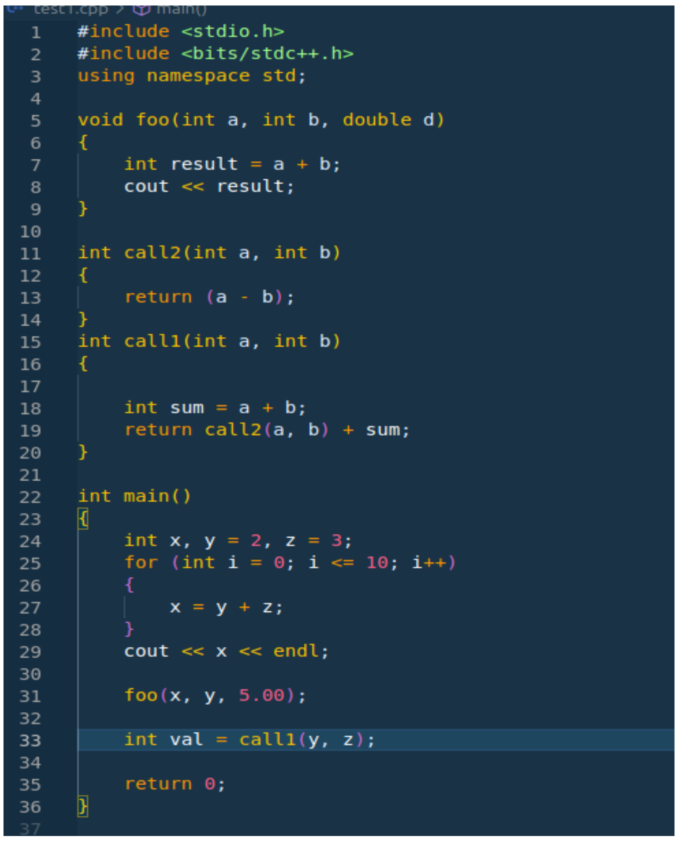
Output



```
ritesh@ritesh-Nitro-AN515-44:~/Desktop/LLVM_Pr
Function: foo(int, int, double)
Basic Block 1
Memory Requirements: 896 bytes
Number of Basic Blocks in this function: 1

Function: main
Basic Block 1
Memory Requirements: 640 bytes
Basic Block 2
Memory Requirements: 192 bytes
Basic Block 3
Memory Requirements: 320 bytes
Basic Block 4
Memory Requirements: 256 bytes
Basic Block 5
Memory Requirements: 448 bytes
Number of Basic Blocks in this function: 5
```

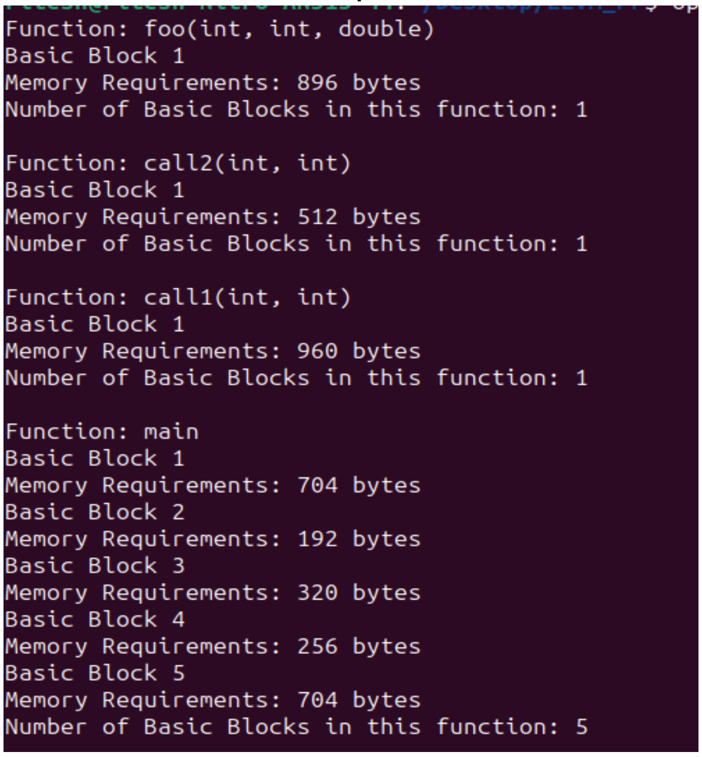
3. For Loop + Another Function

Code



```cpp
#include <stdio.h>
#include <bits/stdc++.h>
using namespace std;

void foo(int a, int b, double d)
{
    int result = a + b;
    cout << result;
}

int call2(int a, int b)
{
    return (a - b);
}
int call1(int a, int b)
{

    int sum = a + b;
    return call2(a, b) + sum;
}

int main()
{
    int x, y = 2, z = 3;
    for (int i = 0; i <= 10; i++)
    {
        x = y + z;
    }
    cout << x << endl;

    foo(x, y, 5.00);

    int val = call1(y, z);

    return 0;
}
```

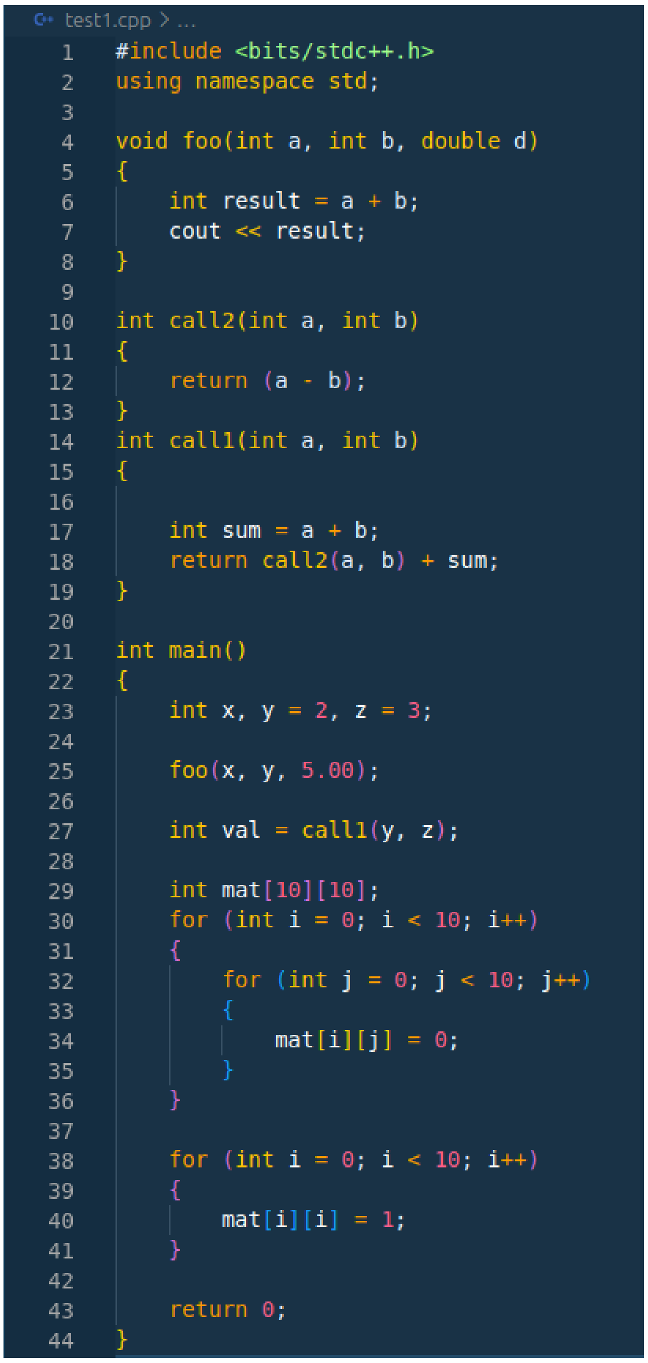
Output



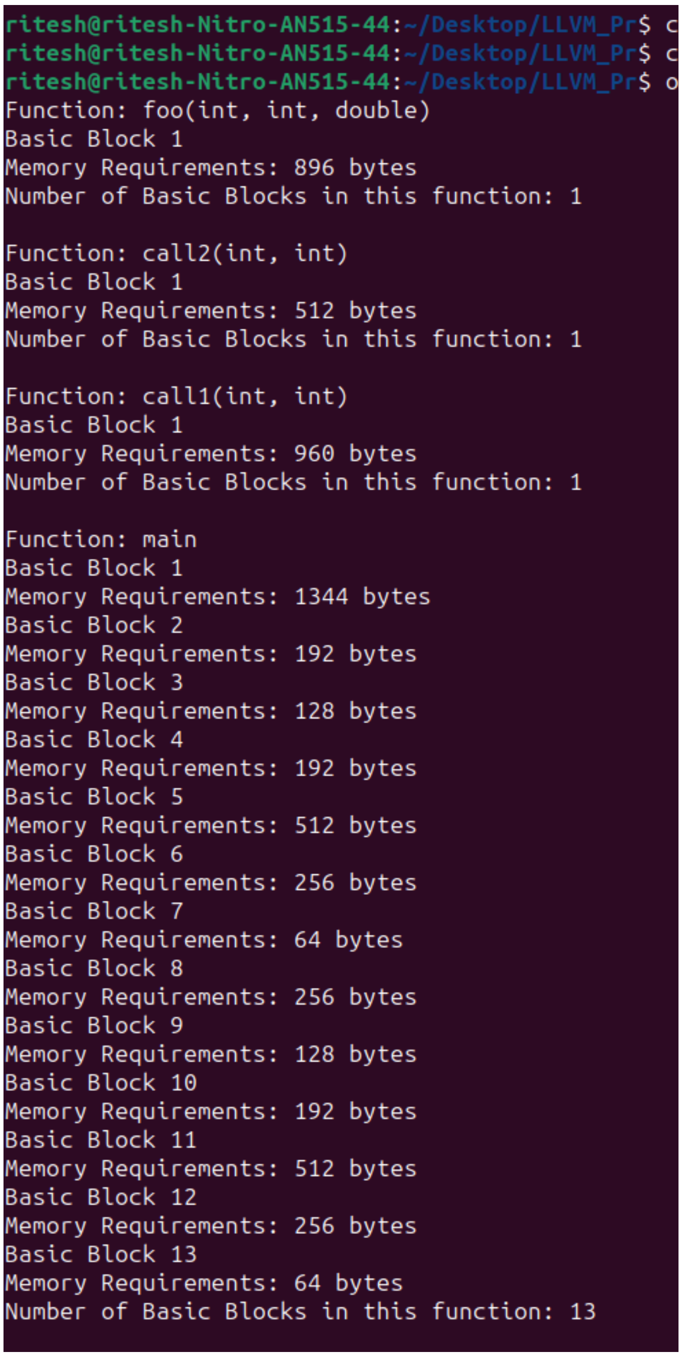
```
Function: foo(int, int, double)
Basic Block 1
Memory Requirements: 896 bytes
Number of Basic Blocks in this function: 1

Function: call2(int, int)
Basic Block 1
Memory Requirements: 512 bytes
Number of Basic Blocks in this function: 1

Function: call1(int, int)
Basic Block 1
Memory Requirements: 960 bytes
Number of Basic Blocks in this function: 1

Function: main
Basic Block 1
Memory Requirements: 704 bytes
Basic Block 2
Memory Requirements: 192 bytes
Basic Block 3
Memory Requirements: 320 bytes
Basic Block 4
Memory Requirements: 256 bytes
Basic Block 5
Memory Requirements: 704 bytes
Number of Basic Blocks in this function: 5
```

4. 10 x 10 Identity Matrix Allocation + 3 Functions

Code and Output



```cpp
#include <bits/stdc++.h>
using namespace std;

void foo(int a, int b, double d)
{
    int result = a + b;
    cout << result;
}

int call2(int a, int b)
{
    return (a - b);
}
int call1(int a, int b)
{

    int sum = a + b;
    return call2(a, b) + sum;
}

int main()
{
    int x, y = 2, z = 3;

    foo(x, y, 5.00);

    int val = call1(y, z);

    int mat[10][10];
    for (int i = 0; i < 10; i++)
    {
        for (int j = 0; j < 10; j++)
        {
            mat[i][j] = 0;
        }
    }

    for (int i = 0; i < 10; i++)
    {
        mat[i][i] = 1;
    }

    return 0;
}
```



```
ritesh@ritesh-Nitro-AN515-44:~/Desktop/LLVM_Pr$ c
ritesh@ritesh-Nitro-AN515-44:~/Desktop/LLVM_Pr$ c
ritesh@ritesh-Nitro-AN515-44:~/Desktop/LLVM_Pr$ o
Function: foo(int, int, double)
Basic Block 1
Memory Requirements: 896 bytes
Number of Basic Blocks in this function: 1

Function: call2(int, int)
Basic Block 1
Memory Requirements: 512 bytes
Number of Basic Blocks in this function: 1

Function: call1(int, int)
Basic Block 1
Memory Requirements: 960 bytes
Number of Basic Blocks in this function: 1

Function: main
Basic Block 1
Memory Requirements: 1344 bytes
Basic Block 2
Memory Requirements: 192 bytes
Basic Block 3
Memory Requirements: 128 bytes
Basic Block 4
Memory Requirements: 192 bytes
Basic Block 5
Memory Requirements: 512 bytes
Basic Block 6
Memory Requirements: 256 bytes
Basic Block 7
Memory Requirements: 64 bytes
Basic Block 8
Memory Requirements: 256 bytes
Basic Block 9
Memory Requirements: 128 bytes
Basic Block 10
Memory Requirements: 192 bytes
Basic Block 11
Memory Requirements: 512 bytes
Basic Block 12
Memory Requirements: 256 bytes
Basic Block 13
Memory Requirements: 64 bytes
Number of Basic Blocks in this function: 13
```