Code for macro pass1

```
#include <iostream>
#include <fstream>
#include <unordered_map>
#include <vector>
#include <sstream>
#include <string>

using namespace std;

// Structure for the Macro Name Table (MNT)
struct MNTEntry {
  string macroName;
  int mdtIndex;
};

// Global variables for tables
unordered_map<string, MNTEntry> MNT;  // Macro Name Table
vector<string> MDT;                   // Macro Definition Table
vector<string> ALA;                   // Argument List Array

// Function to split a line into words
vector<string> splitLine(const string& line) {
  vector<string> tokens;
  stringstream ss(line);
  string word;
  while (ss >> word) {
    tokens.push_back(word);
  }
  return tokens;
}

// Function to process the macro definition and add to MNT, MDT, and ALA
void processMacro(ifstream &input, const string& macroName) {
  MNTEntry mntEntry;
  mntEntry.macroName = macroName;
  mntEntry.mdtIndex = MDT.size(); // Store the start of macro in MDT
  MNT[macroName] = mntEntry;

  string line;
  vector<string> tokens;

  // Read the argument list for the macro (if any)
  getline(input, line);
  tokens = splitLine(line);
  for (const string& token : tokens) {
    if (token != "MACRO") {
      ALA.push_back(token);  // Store arguments in ALA
    }
  }

  // Process the body of the macro until MEND
  while (getline(input, line)) {
    tokens = splitLine(line);
    if (tokens[0] == "MEND") {
      MDT.push_back("MEND");
```

```
            break;
        } else {
            MDT.push_back(line); // Add line to MDT
        }
    }
}

// Function to process the input assembly file for Pass 1
void macroPass1(const string& inputFileName) {
    ifstream inputFile(inputFileName);
    if (!inputFile.is_open()) {
        cerr << "Error opening input file!" << endl;
        return;
    }

    string line;
    vector<string> tokens;

    // Read the file line by line
    while (getline(inputFile, line)) {
        tokens = splitLine(line);
        if (tokens.empty()) continue;

        // If MACRO keyword is found, process the macro
        if (tokens[0] == "MACRO") {
            if (tokens.size() > 1) {
                processMacro(inputFile, tokens[1]);
            }
        }
    }

    inputFile.close();
}

// Function to write the MNT, MDT, and ALA to output files
void writeTables() {
    ofstream mntFile("MNT.txt"), mdtFile("MDT.txt"), alaFile("ALA.txt");

    // Write MNT
    for (const auto& entry : MNT) {
        mntFile << entry.first << " " << entry.second.mdtIndex << endl;
    }

    // Write MDT
    for (size_t i = 0; i < MDT.size(); ++i) {
        mdtFile << i << " " << MDT[i] << endl;
    }

    // Write ALA
    for (size_t i = 0; i < ALA.size(); ++i) {
        alaFile << i << " " << ALA[i] << endl;
    }

    mntFile.close();
    mdtFile.close();
    alaFile.close();
}
```

```
int main() {
    string inputFileName;

    // Prompt for input file (assembly file)
    cout << "Enter the input assembly file (e.g., input.asm): ";
    cin >> inputFileName;

    // Perform Macro Pass 1
    macroPass1(inputFileName);

    // Write the tables to respective files
    writeTables();

    cout << "Macro Pass 1 completed. Check MNT.txt, MDT.txt, and ALA.txt files for output." << endl;

    return 0;
}
```