```
#include <iostream>
#include "RecordSet.h"
namespace sdds {

   RecordSet::RecordSet() {
      m_string = nullptr;
      m_string_count = 0;
   }

   RecordSet::~RecordSet() {
      delete[] m_string;
      m_string = nullptr;
   }

   RecordSet::RecordSet(const RecordSet &src) {
      *this = src;
   }

   RecordSet &RecordSet::operator=(const RecordSet &src) {

      if(this != &src)
      {
         delete[] m_string;
         m_string = nullptr;
         m_string_count = src.m_string_count;
         m_string = new string[m_string_count];
         for(int i=0;i<m_string_count;i++)
        {
            m_string[i] = src.m_string[i];
        }
      }
      return *this;
   }

   size_t RecordSet::size() {
      return m_string_count;
   }

   string RecordSet::getRecord(size_t index) {
      if(m_string == nullptr)
      {
         return "";
      }
      else{
```

```
        return m_string[index];
      }
    }

    RecordSet::RecordSet(char *File) {
      ifstream file(File);
      string temp;
      int i=0;
      while(!file.eof())
      {
        getline(file,temp,' ');
        i++;
      }
      m_string =new string[i];
      i=0;
      file.seekg(0);
      while(!file.eof())
      {
        getline(file,m_string[i],' ');
        i++;
      }
      m_string_count = i;
      file.close();
    }

    RecordSet::RecordSet(RecordSet&& src) noexcept {
      operator=(move(src));
    }

    RecordSet& RecordSet::operator=(RecordSet&& src)  noexcept {
      if(this != &src)
      {
        m_string = src.m_string;
        src.m_string = nullptr;
        m_string_count = src.m_string_count;
        src.m_string_count = 0;
      }
      return *this;
    }

}
```