EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

```
#include <iostream>
#include <string>

using namespace std;

class Student
{
private:
   char * studentName;
public:
  Student(const char* name)
  {
    cout << "Inside constructor: passed in string = " << name;
    studentName = new char[50];
    strcpy(studentName,name);
    cout << "Initialized string to" << studentName << endl;
  }

  ~Student()
  {
    cout << "Freeing up memory for string " << studentName << endl;
    delete[] studentName;
  }

};
```

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

```
#include <iostream>
#include <string>

using namespace std;

class Student
{
private:
   char * studentName;
public:
  Student(const char* name)
  {
    cout << "Inside constructor: passed in string = " << name;
    studentName = new char[50];
    strcpy(studentName,name);
    cout << "Initialized string to" << studentName << endl;
  }

  ~Student()
  {
    cout << "Freeing up memory for string " << studentName << endl;
    delete[] studentName;
  }

};
```

The destructor

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

```
#include <iostream>
#include <string>

using namespace std;

class Student
{
private:
   char * studentName;
public:
  Student(const char* name)
  {
    cout << "Inside constructor: passed in string = " << name;
    studentName = new char[50];
    strcpy(studentName,name);
    cout << "Initialized string to" << studentName << endl;
  }

  ~Student()
  {
    cout << "Freeing up memory for string " << studentName << endl;
    delete[] studentName;
  }

};
```

The destructor has the same name as the class, but with a preceding tilde (~)

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

```
#include <iostream>
#include <string>

using namespace std;

class Student
{
private:
   char * studentName;
public:
  Student(const char* name)
  {
    cout << "Inside constructor: passed in string = " << name;
    studentName = new char[50];
    strcpy(studentName,name);
    cout << "Initialized string to" << studentName << endl;
  }

  ~Student()
  {
    cout << "Freeing up memory for string " << studentName << endl;
    delete[] studentName;
  }

};
```

The destructor frees up the memory that was allocated for the member variable studentName

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

```
#include <iostream>
#include <string>

using namespace std;

class Student
{
private:
   char * studentName;
public:
  Student(const char* name)
  {
    cout << "Inside constructor: passed in string = " << name;
    studentName = new char[50];
    strcpy(studentName,name);
    cout << "Initialized string to" << studentName << endl;
  }

  ~Student()
  {
    cout << "Freeing up memory for string " << studentName << endl;
    delete[] studentName;
  }

};
```

This memory was allocated inside the constructor, and it was freed in the destructor

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

```
#include <iostream>
#include <string>

using namespace std;

class Student
{
private:
    char * studentName;
public:
   Student(const char* name)
   {
     cout << "Inside constructor: passed in string = " << name;
     studentName = new char[50];
     strcpy(studentName,name);
     cout << "Initialized string to" << studentName << endl;
   }

   ~Student()
   {
     cout << "Freeing up memory for string " << studentName << endl;
     delete[] studentName;
   }

};
```

BTW, new and delete are the C++ versions of malloc and free. More on them in just a bit.

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

```
   ~Student()
   {
     cout << "Freeing up memory for string " << studentName <<
endl;
     delete[] studentName;
   }

};

int main()
{
   const char name[50] = "Vitthal";
   Student student(name);
   student.print();
   cout << "Exiting the program - last line of code. Bye!"<<endl;
   return 0;
}
```

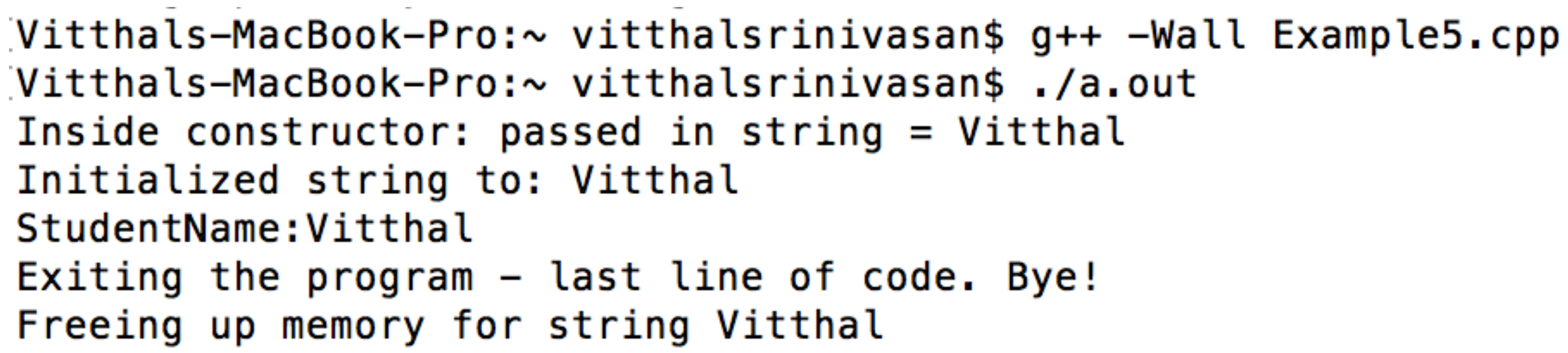
Now, run it and see what happens!

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

Now, run it and see what happens!



```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example5.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Inside constructor: passed in string = Vitthal
Initialized string to: Vitthal
StudentName:Vitthal
Exiting the program - last line of code. Bye!
Freeing up memory for string Vitthal
```

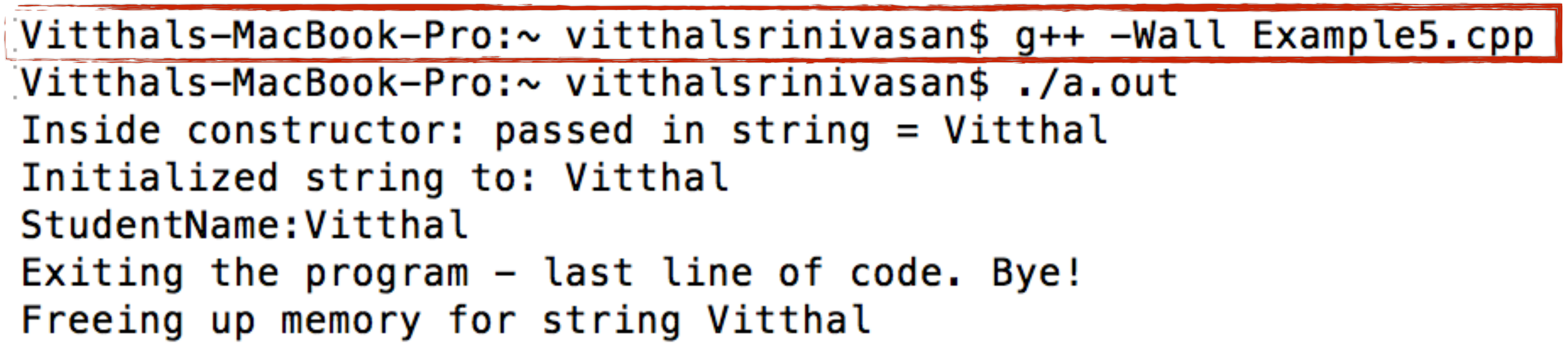
EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

Now, run it and see what happens!

Compile the code first



```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example5.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Inside constructor: passed in string = Vitthal
Initialized string to: Vitthal
StudentName:Vitthal
Exiting the program - last line of code. Bye!
Freeing up memory for string Vitthal
```

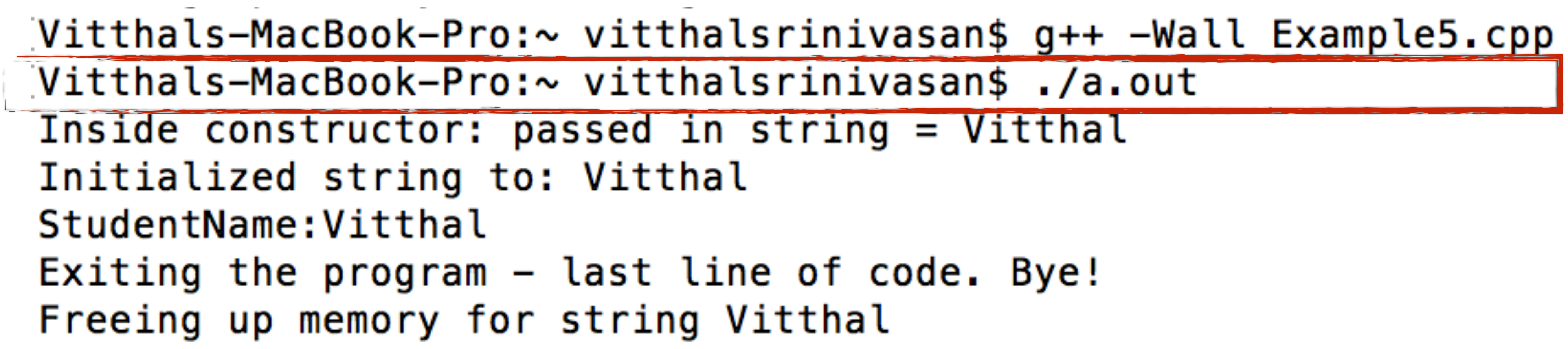
EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

Now, run it and see what happens!

Then run it



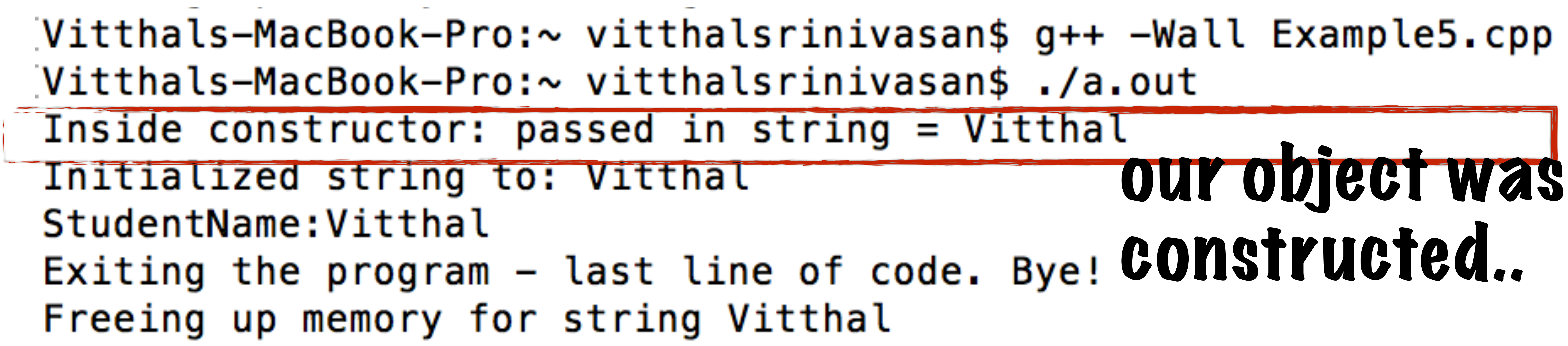
```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example5.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Inside constructor: passed in string = Vitthal
Initialized string to: Vitthal
StudentName:Vitthal
Exiting the program - last line of code. Bye!
Freeing up memory for string Vitthal
```

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

Now, run it and see what happens!



```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example5.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Inside constructor: passed in string = Vitthal
Initialized string to: Vitthal
StudentName:Vitthal
Exiting the program - last line of code. Bye!
Freeing up memory for string Vitthal
```

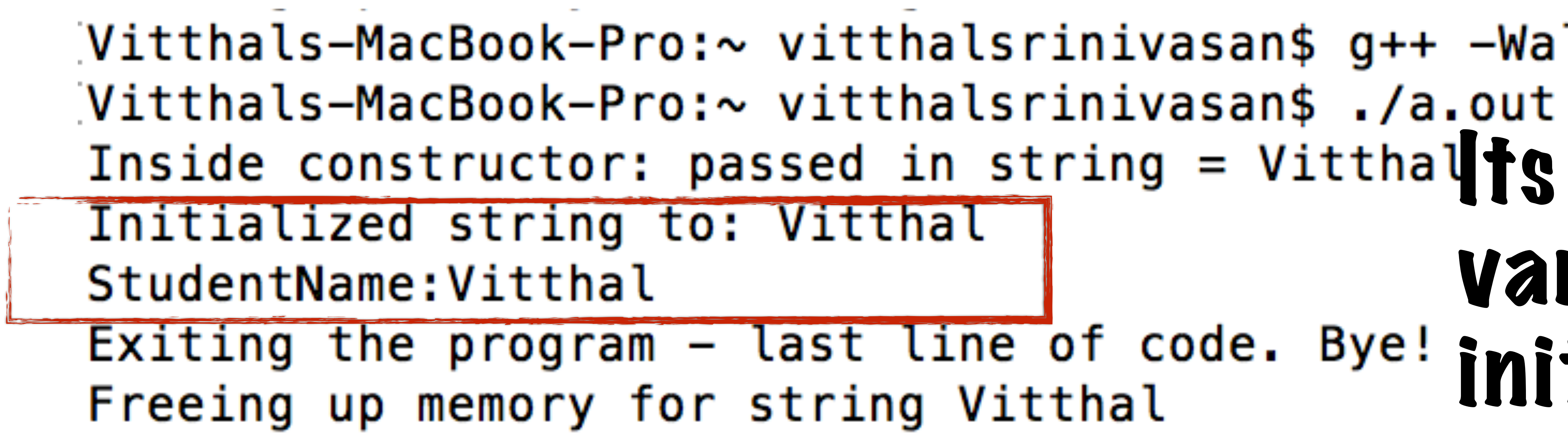
our object was constructed..

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

Now, run it and see what happens!



```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example5.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Inside constructor: passed in string = Vitthal
Initialized string to: Vitthal
StudentName:Vitthal
Exiting the program - last line of code. Bye!
Freeing up memory for string Vitthal
```

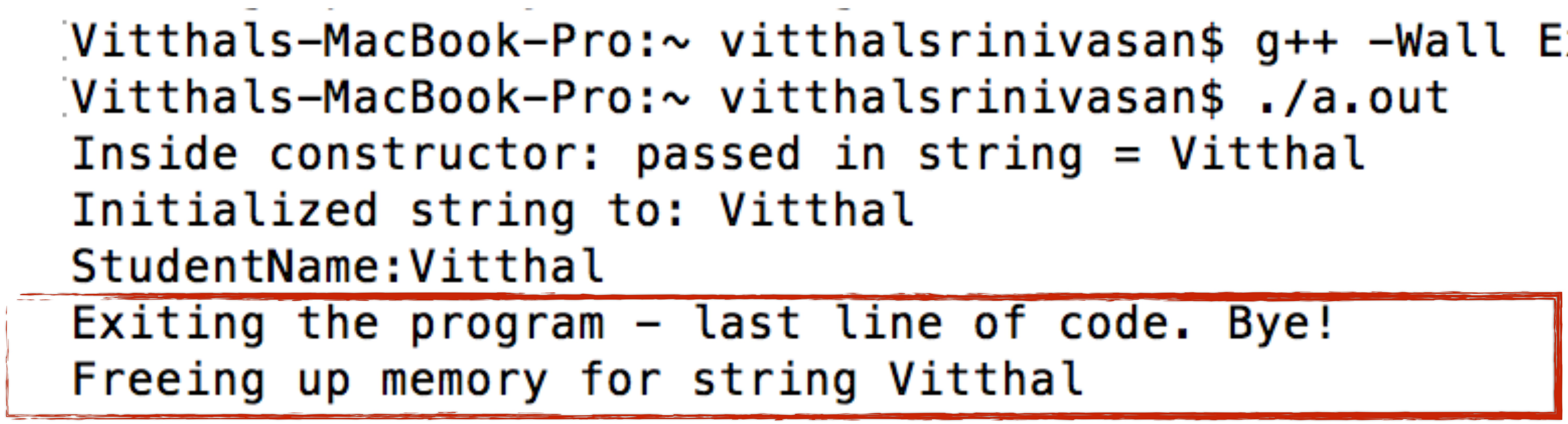
Its char* member variable was initialised

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

Now, run it and see what happens!



```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example5.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Inside constructor: passed in string = Vitthal
Initialized string to: Vitthal
StudentName:Vitthal
Exiting the program - last line of code. Bye!
Freeing up memory for string Vitthal
```

But what's this? The memory is freed up after the last line of code is executed!

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

```
  ~Student()
  {
    cout << "Freeing up memory for string " << studentName <<
endl;
    delete[] studentName;
  }

};
```

Now, run it and see what happens!

```
int main()
{
  const char name[50] = "Vitthal";
  Student student(name);
  student.print();
  cout << "Exiting the program - last line of code. Bye!"<<endl;
  return 0;
}
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example5.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Inside constructor: passed in string = Vitthal
Initialized string to: Vitthal
StudentName:Vitthal
Exiting the program - last line of code. Bye!
Freeing up memory for string Vitthal
```

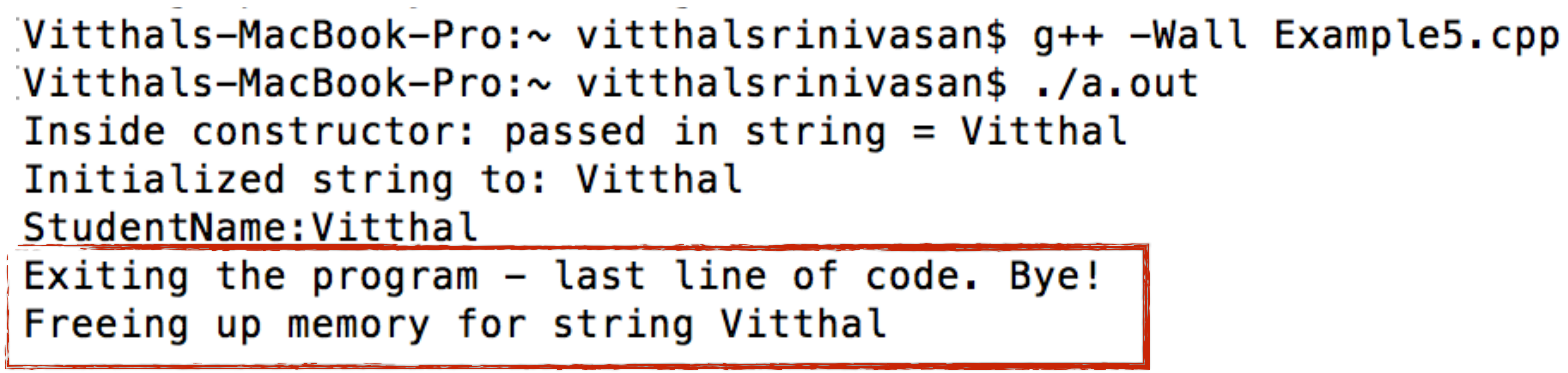
But what's this? The memory is freed up after the last line of code is executed!

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

Now, run it and see what happens!



```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example5.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Inside constructor: passed in string = Vitthal
Initialized string to: Vitthal
StudentName:Vitthal
Exiting the program - last line of code. Bye!
Freeing up memory for string Vitthal
```

the destructor got called AFTER the last line of code was executed. In other words, after the closing brace of the main function!

EXAMPLE 5: CORRECTLY USE A DESTRUCTOR TO CLEAN UP MEMBER VARIABLES

A DESTRUCTOR IS A SPECIAL MEMBER FUNCTION THAT CLEANS UP MEMBER VARIABLES OF AN OBJECT

A DESTRUCTOR HAS THE SAME NAME AS THE CLASS, BUT PRECEDED BY A TILDE (~)

DESTRUCTORS ARE CRUCIAL WHEN YOUR CLASS HAS POINTERS OR FILE HANDLES AMONG ITS MEMBER VARIABLES.

In such cases, not freeing memory or closing files can lead to serious bugs - and memory and resource leaks.