A system to teach Introductory Programming
People...

- Rajdeep Das (IITK)
- Umair Z Ahmed (IITK)
- Naman Bansal (IITK)
- Ivan Radiček (VUT)
- Several Interns
- Sumit Gulwani (MSR, IITK)
- Amey Karkare (IITK)
- More students are added...
Challenges

- Teaching 1st level programming course is difficult
- Specially in India
  - Lack of proper infrastructure
  - Many Languages (not the programming languages)
- Wide gap in the level of exposure to computers
  - Competitive programmers to Not-even-seen-a-computer-yet!
Major Challenges for us

- How to keep good programmers engaged while teaching simple stuff to beginners?
- How to provide early feedback?
- Without too many extra resources
  - Limited TA support
  - Varying expertise of TAs
  - Nobody likes to work extra hours 😞
- Use of existing computers
  - Various flavors of OSes
Solution?

Code Hunt?
- Talk with Judith in Sept 2013 to use it at IIT Kanpur
- Did not work out (for several different reasons)

IDEone?
- Browser based IDE
- Nothing else!
Setup

- ESC101 @ IIT Kanpur
- Approx 400 students every semester
- Weekly Load
  - for student: 3 Lectures, 1 Tutorial, 1 Lab
  - for instructor: 3 Lectures, 1 Tutorial (summarize the week, do-s and don’t-s, quizzes, ...), 4 Labs (?)

- Other courses have labs too!
  - Space and time table issues => class divided into 4 groups with different lab days
  - ESC101 lab on M, Tu, W, Th
The System : Finally

Its demo time
Student Interface
Dashboard

Ongoing Event

TEST-1
Ends on Sat Feb 21 2015 at 12:00:00

<table>
<thead>
<tr>
<th>Q1</th>
<th>20 Points</th>
<th>not-submitted</th>
<th>⬇️ Start Coding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2</td>
<td>20 Points</td>
<td>not-submitted</td>
<td>⬇️ Start Coding</td>
</tr>
<tr>
<td>Q3</td>
<td>20 Points</td>
<td>not-submitted</td>
<td>⬇️ Start Coding</td>
</tr>
</tbody>
</table>

Course Statistics

Course Problems

- Submitted: 0
- Not Submitted: 43

Course Events

- Labs: 5
- Exams: 2
- Quizzes: 0

GRADE CARD
TEST-1 - Q1 (20 Points)

Problem Statement

Write a program to check whether a given number is a **Palindrome** or not. An **Palindrome** number is the one in which the reverse of the number is equal to the number itself.

**Input:** An int, i.e. the number \( n \) (\( n > 0 \))

**Output:**

YES if the given number is an **Palindrome**.
NO otherwise.

**EXAMPLES**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>17371</td>
<td>YES</td>
</tr>
<tr>
<td>1551</td>
<td>YES</td>
</tr>
<tr>
<td>42</td>
<td>NO</td>
</tr>
</tbody>
</table>
File Menu
Problem Statement

Write a program to check whether a given number is a Palindrome or not. A Palindrome number is the one in which the reverse of the number is equal to the number itself.

**Input:** An int, i.e. the number $n$ ($n > 0$)

**Output:**
- YES if the given number is a Palindrome.
- NO otherwise.

**Examples**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>17371</td>
<td>YES</td>
</tr>
<tr>
<td>1551</td>
<td>YES</td>
</tr>
<tr>
<td>42</td>
<td>NO</td>
</tr>
</tbody>
</table>
TEST-1 - Q1 (20 Points)

Problem Statement

Write a program to check whether a given number is a Palindrome or not. An Palindrome number is the one in which the reverse of the number is equal to the number itself.

Input: An int, i.e. the number n (n > 0)
Output:
YES if the given number is an Palindrome.
NO otherwise.

EXAMPLES

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>17371</td>
<td>YES</td>
</tr>
<tr>
<td>1551</td>
<td>YES</td>
</tr>
<tr>
<td>42</td>
<td>NO</td>
</tr>
</tbody>
</table>

```c
#include<stdio.h>

int main()
{
    int num;
    scanf("%d", &num);
    return 0;
}
```

Compilation failed.
program.c: In function 'main':
program.c:5:10: error: 'num' undeclared (first use in this function)
    scanf("%d", &num);
  ^
program.c:5:10: note: each undeclared identifier is reported only once for each function it appears in
Run → Compile

TEST-1 - Q1 (20 Points)

Problem Statement

Write a program to check whether a given number is a Palindrome or not. A Palindrome number is the one in which the reverse of the number is equal to the number itself.

Input: An int, i.e. the number \( n \) \( (n > 0) \)
Output:
YES if the given number is a Palindrome.
NO otherwise.

EXAMPLES

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>17371</td>
<td>YES</td>
</tr>
<tr>
<td>1551</td>
<td>YES</td>
</tr>
<tr>
<td>42</td>
<td>NO</td>
</tr>
</tbody>
</table>

Your program did NOT compile successfully! Please check the annotations on the editor and/or the compiler messages in the console.
```c
#include<stdio.h>

int main()
{
    int num;
    scanf("%d", &num);
}
```

You are using a variable over here that is undeclared. 'num' has not been declared before. Please declare this variable as something like 'int num;' or 'int num = 0;'.

Compilation failed.
program.o: in function 'main':
program.o:5:18: error: 'num' undeclared (first use in this function)
    scanf("%d", &num);
program.o:5:18: note: each undeclared identifier is reported only once for each function it appears in
TEST-1 - Q1 (20 Points)

Problem Statement

Write a program to check whether a given number is a Palindrome or not. An Palindrome number is the one in which the reverse of the number is equal to the number itself.

Input: An int, i.e. the number n (n > 0).
Output:
YES if the given number is an Palindrome.
NO otherwise.

EXAMPLES
Input  Output
17371  YES
1551  YES
42  NO

When you use format specifiers such as '%d', the type of corresponding variable must match. It is also likely that you forgot to use addressof (&) in scanf.

You cannot use a 'int' variable over here as '%d' requires a 'int *' type.
You should always initialize a variable before you use it in your program. Uninitialized variables tend to contain garbage values and your program may not produce the output that you desired. The variable 'num' is uninitialized over here. Set it to something like 'num=0'.

#include<stdio.h>
int main()
{
    int num;
    scanf("%d", &num);
    // Rest of the code
}
TEST-1 - Q1 (20 Points)

Ends in 11 days

Problem Statement

Write a program to check whether a given number is a Palindrome or not. A Palindrome number is the one in which the reverse of the number is equal to the number itself.

Input: An int, i.e. the number n (n > 0)
Output:
YES if the given number is a Palindrome.
NO otherwise.

EXAMPLES
Input  Output
17371  YES
1551   YES
42     NO

When you use format specifiers such as '%d', the type of corresponding variable must match. It is also likely that you forgot to use addressof (&) in scanf. You cannot use a 'float *' variable over here as '%d' requires a 'int *' type.
Run → Evaluate; Feedback

Tutor

2:14:05 PM

1. You should try to use a loop (for or while) to compute reverse number.

2:14:05 PM

Evaluation Results

<table>
<thead>
<tr>
<th>#</th>
<th>INPUT</th>
<th>EXPECTED OUTPUT</th>
<th>ACTUAL OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17371</td>
<td>YES</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>1551</td>
<td>YES</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>42</td>
<td>NO</td>
<td>X</td>
</tr>
</tbody>
</table>

Your program passed 0 out of 2 hidden test case(s).
NOTE: These may not be the only hidden test cases that your program be evaluated upon.

You have feedback on your program. Look at the tutor tab to view the feedback.

Your program did not pass all test cases. Please view the tutor panel to get details.
TEST-1 - Q1 (20 Points)

Problem Statement

Write a program to check whether a given number is a Palindrome or not. A Palindrome number is the one in which the reverse of the number is equal to the number itself.

Input: An int, i.e. the number n (n > 0)
Output:
YES if the given number is a Palindrome.
NO otherwise.

Examples

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>17371</td>
<td>YES</td>
</tr>
<tr>
<td>1551</td>
<td>YES</td>
</tr>
<tr>
<td>42</td>
<td>NO</td>
</tr>
</tbody>
</table>

Compilation failed.
program.c: In function 'main':
program.c:5:18: error: 'num' undeclared (first use in this function)
    scanf("%d", &num);
^

program.c:5:18: note: each undeclared identifier is reported only once for each function it appears in
TEST-1 - Q1 (20 Points)

Problem Statement

Write a program to print last two digits of a number in reverse. Assume the number to be \( \geq 10 \).

Input: An int, i.e., the number \( n (n \geq 10) \).

Output: the reverse of last two digits.

EXAMPLES

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>43</td>
</tr>
<tr>
<td>1551</td>
<td>15</td>
</tr>
<tr>
<td>42</td>
<td>24</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
</tr>
</tbody>
</table>
Feedback

Tutor

2:23:43 PM

1. Add assignment to 'd' to BEGINNING of main function.
2. Check loop condition at line 7.
3. Add assignment to 'r' to BEGINNING of loop starting at line 7.
4. Add assignment to 'n' to BEGINNING of loop starting at line 7.
5. Check 'printf(s) AFTER loop starting at line 7.

2:23:43 PM

Evaluation Results

<table>
<thead>
<tr>
<th>#</th>
<th>INPUT</th>
<th>EXPECTED OUTPUT</th>
<th>ACTUAL OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17371</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1551</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>42</td>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>

Your program passed 0 out of 2 hidden test

#include <stdio.h>

int main()
{
    int n, d, r;
    r = 0;
    scanf("%d", &n);
    while (n >= 0) {
        }
        if (n == r) printf("YES");
        else printf("NO");
        return 0;
    

Compilation failed.
program.c: In function 'main':
program.c:5:18: error: 'num' undeclared (first use in this function)
scanf("%d", &num);

program.c:5:18: note: each undeclared identifier is reported only once for each function it appears in
Tutor

2:26:57 PM

1. Use new (int) variable.
2. Check assignment to "i" at line 10.
3. Check 'printf(s) AFTER' loop starting at line 7.

2:26:57 PM

Evaluation Results

<table>
<thead>
<tr>
<th>#</th>
<th>INPUT</th>
<th>EXPECTED OUTPUT</th>
<th>ACTUAL OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17371</td>
<td>YES</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>1551</td>
<td>YES</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>42</td>
<td>NO</td>
<td>no</td>
</tr>
</tbody>
</table>

Your program passed 0 out of 2 hidden test

#include <stdio.h>

int main()
{
    int n, d, r;
    r = 0;
    scanf("%d", &n);
    while (n > 0) {
        d = n%10;
        n = n/10;
        r = r*10 + d;
    }
    if (n == r) printf("YES");
    else printf("NO");
    return 0;
}
Even more Feedback

Tutor
2:28:55 PM
1. Check assignment to 'r' at line 11
2. Check 'printf(s) AFTER loop starting at line 8.

Evaluation Results

<table>
<thead>
<tr>
<th>#</th>
<th>INPUT</th>
<th>EXPECTED OUTPUT</th>
<th>ACTUAL OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17371</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>2</td>
<td>1551</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>3</td>
<td>42</td>
<td>NO</td>
<td>no</td>
</tr>
</tbody>
</table>

Your program passed 2 out of 2 hidden test case(s).
NOTE: These may not be the only hidden test cases that your program be evaluated upon.

#include <stdio.h>

int main() {
    int saved, n, d, r;
    r = 0;
    scanf("%d", &n);
    saved = n;
    while (n > 0) {
        d = n%10;
        n = n/10;
        r = r*10 + d;
    }

    if (saved == r) printf("YES");
    else printf("no");

    return 0;
}

Compilation failed.
Program.c: In function 'main':
program.c:5:10: error: 'num' undeclared (first use in this function)
    scanf("%d", &num);

program.c:5:11: note: each undeclared identifier is reported only once for each function it appears in
Success

Congratulations! Your program has passed all test cases. You can now submit this program.

Your program passed 2 out of 2 hidden test case(s).
NOTE: These may not be the only hidden test cases that your program will be evaluated upon.

1. Check assignment to 'y' at line 11.
2. Check 'printf(s' AFTER loop starting at line 8.

Compilation failed.
program.c: In function 'main':
program.c:5:18: error: 'num' undeclared (first use in this function)
    scanf("%d", &num);

program.c:5:18: note: each undeclared identifier is reported only once for each function it appears in
Admin Interface
Problem Management

Select A Problem From The Left

Create a Problem
Local (PVD) Identifier
  eg: p1v1d1
Category
  eg: Strings

LAB-0 (INTRO)
LAB-1 (IO)
LAB-2 (CONDITIONALS)
  P1-V1-D1 (Mon-Vowels)
  P1-V2-D1 (Mon-Interval)
  P2-V1-D1 (Tue-Equality)
  P2-V2-D1 (Tue-2ndMax)
  P3-V1-D1 (Wed-ObtuseTriangle)
  P3-V2-D1 (Wed-MidChar)
  P4-V1-D1 (Thu-RightTriangle)
Problem Statement

Write a C Program that takes as input a long integer \( k > 0 \) and prints the integer \( n > 0 \) such that \( m! \leq k < (n+1)! \). Here \( m! \) means factorial of positive integer \( m \).

Note: Use long data type to store the value of \( k \) and \( m! \).

Examples:

Input:
1
Output:
1

Input:
5
Output:
2

Input:
1080
Instructor’s Solution

Primary Solution Code

```c
#include<stdio.h>

int main()
{
    int n=1;
    long k, fact = 1;
    scanf("%ld", &k);

    while (k >= fact) {
        n = n+1;
        fact = fact * n;
    }

    printf("%d
", n-1);
    return 0;
}
```

Solution Specifications

Solution Initial Template

```c
#include <stdio.h>
```
Specifications and Template

Solution Specifications

Initial Template

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    // Fill this area with your code.
    return 0;
}
```

TEST CASES

AUTOMATICALLY GENERATED TEST CASES

This problem has no automated test cases. Add automated/pre-generated test cases to this problem.

MANUALLY ADDED TEST CASES

Add a test case manually
Testcases: Automated & Manual

#include <stdio.h>
#include <stdlib.h>

int main() {
    // Fill this area with your code.
    return 0;
}

TEST CASES

AUTOMATICALLY GENERATED TEST CASES
This problem has no automated test cases. Add automated/pre-generated test cases to this problem.

MANUALLY ADDED TEST CASES

Add a test case manually

<table>
<thead>
<tr>
<th>#</th>
<th>INPUT</th>
<th>OUTPUT</th>
<th>VISIBILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>visible</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>2</td>
<td>visible</td>
</tr>
<tr>
<td>3</td>
<td>1080</td>
<td>6</td>
<td>visible</td>
</tr>
<tr>
<td>4</td>
<td>523456723</td>
<td>12</td>
<td>visible</td>
</tr>
<tr>
<td>5</td>
<td>6227028800</td>
<td>13</td>
<td>invisible</td>
</tr>
<tr>
<td>6</td>
<td>20922789888000</td>
<td>16</td>
<td>invisible</td>
</tr>
</tbody>
</table>
CONTROL PANEL

GCC Compiler Options

Current Flags
- -static
- -g
- -Wall
- -lm
- -O1
- -unsigned-char
- -Wno-unused-result

PLUGINS

Strategy Based Feedback
Author: Ivan Radíček
Enabled

Engine Delays

Compilation:
500 milli-seconds

Execution:
500 milli-seconds

Evaluation:
500 milli-seconds

Execution Sandbox Options

Quotas
Time:
10000 milli-seconds

Memory:
8388608 bytes

Update Quotas
This section shows the syntactic errors which occur in the programs along with their frequencies and other details. The data has been collected from compiler outputs. The right hand side shows the instances of the selected error.

<table>
<thead>
<tr>
<th>#</th>
<th>Type</th>
<th>Class</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>:X1 undeclared</td>
<td>(first use in this function)</td>
<td>7498</td>
</tr>
<tr>
<td>2</td>
<td>unused variable</td>
<td>:X1 [-Wunused-variable]</td>
<td>7190</td>
</tr>
<tr>
<td>3</td>
<td>:X1 is used</td>
<td>uninitialized in this function [-Wuninitialized]</td>
<td>6761</td>
</tr>
<tr>
<td>4</td>
<td>format</td>
<td>:X1 expects argument of type :X2, but argument 2 has type :X3 [-Wformat]</td>
<td>6149</td>
</tr>
<tr>
<td>5</td>
<td>expected</td>
<td>:X1 before :X2 token</td>
<td>4473</td>
</tr>
<tr>
<td>6</td>
<td>control reaches</td>
<td>end of non-void function [-Wreturn-type]</td>
<td>4217</td>
</tr>
<tr>
<td>7</td>
<td>each undeclared</td>
<td>identifier is reported only once for each function it appears in</td>
<td>4160</td>
</tr>
<tr>
<td>8</td>
<td>expected</td>
<td>:X1 before :X2</td>
<td>3934</td>
</tr>
<tr>
<td>9</td>
<td>variable :X1 set</td>
<td>but not used [-Wunused-but-set-variable]</td>
<td>2604</td>
</tr>
<tr>
<td>10</td>
<td>expected expression</td>
<td>before :X1 token</td>
<td>2346</td>
</tr>
</tbody>
</table>
Syntactic Feedback

DataViZ > Syntactic Analysis

This section shows the syntactic errors which occur in the programs along with their frequencies and other details. The data has been collected from compiler outputs. The right hand side shows the instances of the selected error.

1. :X1 undeclared (first use in this function) | 7498
2. unused variable :X1 [Wunused-variable] | 7190
3. :X1 is used uninitialized in this function [Wuninitialized] | 6761
4. format: :X1 expects argument of type :X2, but argument 2 has type :X3 [Wformat=] | 6149
5. expected :X1 before :X2 token | 4473
6. control reaches end of non-void function [Wreturn-type] | 4217
7. each undeclared identifier is reported only once for each function it appears in | 4160
8. expected :X1 before :X2 | 3984
9. variable :X1 set but not used [Wunused-but-set-variable] | 2604
10. expected expression before :X1 token | 2346

Feedback

You should always initialize a variable before you use it in your program. Uninitialized variables tend to contain garbage values and your program may not produce the output that you desired. The variable ':X1' is uninitialized over here. Set it to something like ':X1=0'.

Update

Assignment

#69337

Show / Hide

Assignment
This section shows the syntactic errors which occur in the programs along with their frequencies and other details. The data has been collected from compiler outputs. The right hand side shows the instances of the selected error.

<table>
<thead>
<tr>
<th>#</th>
<th>Type</th>
<th>Error Description</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>❌</td>
<td>X1 undeclared (first use in this function)</td>
<td>7498</td>
</tr>
<tr>
<td>2</td>
<td>❗</td>
<td>unused variable 'X1' [-Wunused-variable]</td>
<td>7199</td>
</tr>
<tr>
<td>3</td>
<td>❗</td>
<td>X1 is used uninitialized in this function [-Wuninitialized]</td>
<td>6761</td>
</tr>
<tr>
<td>4</td>
<td>❗</td>
<td>format 'X1' expects argument of type 'X2, but argument 2 has type 'X3' [-Wformat=-]</td>
<td>6149</td>
</tr>
<tr>
<td>5</td>
<td>❌</td>
<td>expected 'X1' before 'X2' token</td>
<td>4473</td>
</tr>
<tr>
<td>6</td>
<td>❗</td>
<td>control reaches end of non-void function [-Wreturn-type]</td>
<td>4217</td>
</tr>
<tr>
<td>7</td>
<td>❌</td>
<td>each undeclared identifier is reported only once for each function it appears in</td>
<td>4160</td>
</tr>
<tr>
<td>8</td>
<td>❌</td>
<td>expected 'X1' before 'X2'</td>
<td>3034</td>
</tr>
<tr>
<td>9</td>
<td>❗</td>
<td>variable 'X1' set but not used [-Wunused-but-set-variable]</td>
<td>2604</td>
</tr>
<tr>
<td>10</td>
<td>❌</td>
<td>expected expression before 'X1' token</td>
<td>2346</td>
</tr>
<tr>
<td>11</td>
<td>❗</td>
<td>X1 may be used uninitialized in this function [-Wmaybe-uninitialized]</td>
<td>2302</td>
</tr>
<tr>
<td>12</td>
<td>❗</td>
<td>suggest parentheses around assignment used as truth value [-Wparentheses]</td>
<td>2124</td>
</tr>
</tbody>
</table>

**Feedback**

You should always initialize a variable before you use it in your program. Uninitialized variables tend to contain garbage values and your program may not produce the output that you desired. The variable "X1" is uninitialized over here. Set it to something like 'X1=0'.

**Update**

**Assignment #69337**

- a' is used uninitialized in this function [-Wuninitialized]

**Assignment #84401**

- 'Celcius' is used uninitialized in this function [-Wuninitialized]
<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>
User Submissions
Grading Panel

Assignment ID: 115501

Fri Feb 06 2015 16:53:25

Submitted

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    long array[1000]; //stores input
    long value[1000];
    long count[1000];

    //inputs
    int n, i;
    scanf("%d", &n);
    for(i=0; i<n; i++)
        scanf("%ld", &array[i]);

    //initial values
    value[0] = array[0];
    count[0] = 1;
    int index = 0; //index for value[] and count[]

    for(i=1; i<n; i++)
    {
        if(array[i] == value[index]) //if successive element is same
            count[index]++;
        else
        {
            index++; //shift
            value[index] = array[i]; //new value
            count[index] = 1; //reset count
        }
    }
}
Analytics per Submission

Assignment #115247
Analytics

Summary

<table>
<thead>
<tr>
<th>Executions</th>
<th>Compilations</th>
<th>Submissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

Code Size Variation | Code Saves | Syntactic Analysis

Code Size Variation
Analytics per Submission

Code Save Progression

Syntactic Analysis
No Data Available.

Compilation Error Progression
Concluding Remarks

- < 9 months old system, Still in experimentation phase
- Framework allows us to plug-and-play different components
  - Compiler: gcc, python, haskell
  - Feedback: Ivan’s strategy based feedback, compiler message rewriting, some ad-hoc scripts
  - Automated Problem Generation: Ad-hoc programs
  - Automated Testcase Generation: KLEE
Future Work

- Lots of Data waiting to be processed
- Many HCI issues to be resolved
- Feedback tool has known limitations
  - Response time
  - False positives
  - Large # of specifications
- Crowdsourcing
  - test cases, peer review, specification generation, ...
Thank you. Questions?
Some References

- Massively Empowered Classroom: Enhancing Technical Education in India (MSR-TR-2013-127)