Prutor

A system to help teach Introductory Programming
People (Core Team)

- Rajdeep Das (M.Tech 2013-2015)
- Amey Karkare
- Sumit Gulwani (MSR, IITK)
People (Plugin developers)

- Umair Z Ahmed (PhD)
- Mtech: Naman Bansal, Praveen Singh, Ziyaan D.
- External: Ivan Radiček (PhD @ VUT)
- Interns: Sanil Jain, Ayush Sekhari, Aayush Mudgal, Garvit Pahal, Harshit, Kunal Kapila, Pallav Agrawal
- More students added every semester
Challenges

- Teaching 1st level programming course is difficult
- Wide gap in the level of exposure to computers in class
  - Competitive-programmers to Not-even-seen-a-computer-yet!
- Uniformity in grading and feedback
Challenges

- 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?

IDEone (Browser based IDE)?
- Limited capabilities: Missing debugger, No evaluation, No feedback

Judge Programs (SPOJ, Codechef, ...)?
- Typically editor-less
- Grading partial submissions is difficult!
- No feedback

CodeHunt?
- Guess the problem statement
- Learning is a side-effect – not suitable for first time programmers
Approx 425 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
Sign In

ITB LDAP ID

Password

Login
Student Interface
Dashboard

### Ongoing Event

**TEST-1**

- **Ends on Sat Feb 21 2015 at 12:00:00**
- **Q1** 20 Points  not-submitted  ![Start Coding](#)
- **Q2** 20 Points  not-submitted  ![Start Coding](#)
- **Q3** 20 Points  not-submitted  ![Start Coding](#)

### Course Statistics

#### Course Problems

- **Submitted**: 0
- **Not Submitted**: 43

#### Course Events

- **Labs**: 4
- **Exams**: 2
- **Quizzes**: 0

---

[Image of the dashboard interface with the mentioned elements highlighted.]
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
Input   Output
17371   YES
1001    NO
42      NO
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
Run Menu

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
1501     YES
42       NO

```c
#include<stdio.h>

int main()
{
    // fill you code here
    return 0;
}
```
Tabs

Legend

✔️ OUTPUT MATCHED
❌ OUTPUT DID NOT MATCH
⏱ TIME LIMIT EXCEEDED
⚠️ RUNTIME ERROR
⚠️ ABNORMAL TERMINATION
⚠️ RESTRICTED FUNCTION CALL
⚠️ MEMORY LIMIT EXCEEDED

```c
#include<stdio.h>

int main()
{
    // fill you code here
    return 0;
}
```

Console Activity Log Input Output Last saved at 1:48:23 PM


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>177</td>
<td>YES</td>
</tr>
<tr>
<td>455</td>
<td>YES</td>
</tr>
<tr>
<td>42</td>
<td>NO</td>
</tr>
</tbody>
</table>

#include<stdio.h>

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

**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>17971</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.
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 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>1561</td>
<td>YES</td>
</tr>
<tr>
<td>42</td>
<td>NO</td>
</tr>
</tbody>
</table>

```
#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';
}
```
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>17971</td>
<td>YES</td>
</tr>
<tr>
<td>1551</td>
<td>YES</td>
</tr>
<tr>
<td>42</td>
<td>NO</td>
</tr>
</tbody>
</table>

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'.
Run → Evaluate; Feedback

Tutor

2:14:05 PM

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

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

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

program.c:18: note: each undeclared identifier is reported only once for each function it appears in
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 'print(s) AFTER loop starting at line 7.

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 tests
More Feedback

Tutor

2:26:57 PM
1. Use new (int) variable.
2. Check assignment to r at line 10.
3. Check 'printf(s) AFTER loop starting at line 7.

2:26:57 PM
Your program passed 0 out of 2 hidden test
Success 😊

Tutor

2:29:25 PM

Evaluation Results

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

2:28:55 PM

1. Check assignment to ‘r’ at line 11.
2. Check ‘printf(s) AFTER loop starting at line 8.

Congratulations! Your program has passed all test cases. You can now submit this program.
Admin Interface
Problem Management

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

Select A Problem From The Left

Intelligent Tutoring System
Manage Problems

Upload Problems
Problem Statement

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

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

Examples:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>1080</td>
<td></td>
</tr>
</tbody>
</table>
#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("%ld\n", n-1);
    return 0;
}
Specifications and Template

Solution Specifications

Add Specification

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>2</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>6227020860</td>
<td>13</td>
<td>invisible</td>
</tr>
<tr>
<td>6</td>
<td>20922789889000</td>
<td>16</td>
<td>invisible</td>
</tr>
</tbody>
</table>
418 out of 1114 tasks pending.

EXAM-Adv_Track
LAB-2
LAB-3
LAB-4
LAB-5
LAB-6
LAB-7
LAB-8
LAB-9
<table>
<thead>
<tr>
<th>Lab-4</th>
<th>GRADING</th>
<th>Perform Task</th>
<th>Sunday, August 23, 2015 at 18:29:59</th>
<th>Pending</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lab-4</td>
<td>GRADING</td>
<td>Perform Task</td>
<td>Sunday, August 23, 2015 at 18:29:59</td>
<td>Pending</td>
</tr>
<tr>
<td>Lab-4</td>
<td>GRADING</td>
<td>Perform Task</td>
<td>Sunday, August 23, 2015 at 18:29:59</td>
<td>Pending</td>
</tr>
<tr>
<td>Lab-4</td>
<td>GRADING</td>
<td>Perform Task</td>
<td>Sunday, August 23, 2015 at 18:29:59</td>
<td>Pending</td>
</tr>
<tr>
<td>Lab-4</td>
<td>GRADING</td>
<td>Perform Task</td>
<td>Sunday, August 23, 2015 at 18:29:59</td>
<td>Pending</td>
</tr>
<tr>
<td>Lab-4</td>
<td>GRADING</td>
<td>Perform Task</td>
<td>Sunday, August 23, 2015 at 18:29:59</td>
<td>Pending</td>
</tr>
<tr>
<td>Lab-4</td>
<td>GRADING</td>
<td>Perform Task</td>
<td>Sunday, August 23, 2015 at 18:29:59</td>
<td>Pending</td>
</tr>
<tr>
<td>Lab-4</td>
<td>GRADING</td>
<td>Perform Task</td>
<td>Sunday, August 23, 2015 at 18:29:59</td>
<td>Pending</td>
</tr>
<tr>
<td>Lab-4</td>
<td>GRADING</td>
<td>Perform Task</td>
<td>Sunday, August 23, 2015 at 18:29:59</td>
<td>Pending</td>
</tr>
<tr>
<td>Lab-4</td>
<td>GRADING</td>
<td>Perform Task</td>
<td>Sunday, August 23, 2015 at 18:29:59</td>
<td>Pending</td>
</tr>
<tr>
<td>Lab-2</td>
<td>GRADING</td>
<td>Review Task</td>
<td>Friday, August 14, 2015 at 18:29:59</td>
<td>Complete</td>
</tr>
<tr>
<td>Lab-2</td>
<td>GRADING</td>
<td>Review Task</td>
<td>Friday, August 14, 2015 at 18:29:59</td>
<td>Complete</td>
</tr>
<tr>
<td>Lab-2</td>
<td>GRADING</td>
<td>Review Task</td>
<td>Friday, August 14, 2015 at 18:29:59</td>
<td>Complete</td>
</tr>
<tr>
<td>Lab-2</td>
<td>GRADING</td>
<td>Review Task</td>
<td>Friday, August 14, 2015 at 18:29:59</td>
<td>Complete</td>
</tr>
<tr>
<td>Lab-2</td>
<td>GRADING</td>
<td>Review Task</td>
<td>Friday, August 14, 2015 at 18:29:59</td>
<td>Complete</td>
</tr>
<tr>
<td>Lab-3</td>
<td>GRADING</td>
<td>Review Task</td>
<td>Sunday, August 16, 2015 at 18:29:59</td>
<td>Complete</td>
</tr>
<tr>
<td>Lab-3</td>
<td>GRADING</td>
<td>Review Task</td>
<td>Sunday, August 16, 2015 at 18:29:59</td>
<td>Complete</td>
</tr>
</tbody>
</table>
```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
                count[index]++;
            else {
                index++; // shift
                value[index] = array[i]; // new value
                count[index] = 1; // reset count
            }
        }
```
Problem Statement

Assignment ID: 343749

Comments by TA

Grading Panel

Code History Panel

Code History Navigator

```c
int m, i = 0, temp, k = 0, j;
char s[500];

for (i = 0; s[i] != '\0'; i++) {
    scanf("%s", s);
}
while (s[i] != '\0') {
    if (s[i] == '\ '){
        j = i;
        for (j = i + 1; s[j] != '\ '; j++) {
            temp = s[j];
            s[j] = s[j + 1];
            s[j + 1] = temp;
            k++;
        }
    }
}
if (k == 0) {
    i = i + 1;
} else {
    i = j + 1;
}
}
printf("%s", s);
return 0;
```
Control Panel and Settings

Compiler Options

Current Flags
- -static
- -Wall
- -lm
- -Wunused-result
- -O
- -Wextra
- -std=gnu99

Feedback Tools

LOGGING
- Enabled
- Disabled

ESC-FEEDBACK
- Enabled
- Disabled

ESCOMPILER
- Enabled
- Disabled

ESCREPAIR
- Enabled
- Disabled

GRADE-PREDICT
- Enabled
- Disabled

Execution Sandbox Options

Quotas

<table>
<thead>
<tr>
<th>Time</th>
<th>mill-seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory</th>
<th>bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>8355605</td>
<td></td>
</tr>
</tbody>
</table>

Engine Delays

Compilation

| 500    | mill-seconds |

Execution

| 500    | mill-seconds |

Evaluation

| 2000   | mill-seconds |

Update Delays
Plugins/External Tools

- Automated Problem Generation
- Upgrade/Downgrade
- Automated Testcase Generation
- Syntactic Feedback
- Semantic Feedback
- Analytics
- Debugger
Problem Generation

- Problems having similar difficulty
  - Different students can be assigned different set of problems

- Problem statement and solution program

- Variations of the problem
  - different difficulty levels

Key Ideas

- Templates for common idioms
- Rules to instantiate templates and combine them
- Difficulty level: based on the rules used
Problem Generation

Topics covered

- Sequences (Simple Loops)
- Patterns (Nested Loops)
- Arrays, Matrices (2D Arrays), Strings
- Simple Recursion
- Simple Pointers

Combination Rules

- Concatenation
- Boolean Operations
- Conditional Execution
Problem Generation: Sample

Example: Input -
6

Output -
1
22
333
4444
55555
666666

*/

#include <stdio.h>

int main()
{
    int row_number, col_number;
    int height;    // input Parameters (Dimensions of the shape)
    scanf("%d",#h=2##height##h#);    // take input

    // loop over the rows of the pattern
    for(row_number=1;row_number<=height;row_number++)
    {
        // for the characters after the initial spaces with col_number starting from 1
        for(col_number=1;col_number<#h=10#row_number##h#;#h=2#col_number++##h#)
        {
            printf("%d",#h=5#row_number%10##h#);    // print output
        }
    }
}
Problem Generation: Variations

Example: Input -
6
Output -
*
**
***
****
*****
******

Example: Input -
6 6
Output -
*******
*******
*******
*******
*******

Example: Input -
5
Output -
1
234
56789
0123456
789012345
1
234
56789
0123456
789012345

Easier   Easiest   Tougher
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
173971  YES
1551    YES
42      NO

Compilation failed.
program.c: In function 'main':
program.c:3:18: error: 'num' undeclared (first use in this function)
  scanf("%d", &num);
  ^

program.c:3:18: note: each undeclared identifier is reported only once for each function it appears in
**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>1661</td>
<td>15</td>
</tr>
<tr>
<td>42</td>
<td>24</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
</tr>
</tbody>
</table>
Testcase Generation

- Generating large number of problems is not effective if you cannot test the solutions
- Manual test generation does not scale
- We used KLEE* to automate the process

# 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.

<table>
<thead>
<tr>
<th>#</th>
<th>Type</th>
<th>Class</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INFO</td>
<td>X1 undeclared (first use in this function)</td>
<td>7496</td>
</tr>
<tr>
<td>2</td>
<td>WARN</td>
<td>unused variable: X1 [-Wunused-variable]</td>
<td>7190</td>
</tr>
<tr>
<td>3</td>
<td>WARN</td>
<td>X1 is used uninitialized in this function [-Wuninitialized]</td>
<td>6761</td>
</tr>
<tr>
<td>4</td>
<td>WARN</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>INFO</td>
<td>expected: X1 before: X2 token</td>
<td>4473</td>
</tr>
<tr>
<td>6</td>
<td>WARN</td>
<td>control reaches end of non-void function [-Wreturn-type]</td>
<td>4217</td>
</tr>
<tr>
<td>7</td>
<td>WARNING</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>INFO</td>
<td>expected: X1 before: X2</td>
<td>3934</td>
</tr>
<tr>
<td>9</td>
<td>WARN</td>
<td>variable: X1 set but not used [-Wunused-but-set-variable]</td>
<td>2604</td>
</tr>
<tr>
<td>10</td>
<td>INFO</td>
<td>expected expression before: X1 token</td>
<td>2346</td>
</tr>
</tbody>
</table>
Syntactic Feedback

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>🤔</td>
<td>X1 undeclared (first use in this function)</td>
<td>7496</td>
</tr>
<tr>
<td>2</td>
<td>🔴</td>
<td>unused variable :X1 [-Wuninitialized-variable]</td>
<td>7190</td>
</tr>
<tr>
<td>3</td>
<td>🔴</td>
<td>:X1 is used uninitialized in this function [-Wuninitialized]</td>
<td>6781</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>3934</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>
</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’.

Assignment

#69337

Show / Hide

Assignment
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

Assignment #2475434

Assignment #84401

Assignment #2481649
Analytics per Submission

Assignment #115247

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
Analytics per Submission

Code Save Progression

Syntactic Analysis
No Data Available.

Compilation Error Progression
Debugging/Tracing Programs

- Encourage students to trace programs
- Debugger to help them step through the program
- Challenge: Server can not be kept busy for long duration
- Client side debugging

Key Idea

- Run and collect information at server
- Replay at client
Debugging/Tracing Programs

Demo
Architecture
Architecture Overview

- HTTP Proxy/Load balancer
- Database Proxy/Load balancer
- Relational Database
- In-Memory Cache
- Web Application
- Engine

- Service Discovery
- Virtualization
- Infrastructure / Hardware
Components & Connectivity

- In-Memory Cache
- Database Proxy
- Relational Database
- Web Application Service
- Engine Service
- HTTP Proxy
Quality Attributes

• **Scalable**
  ◦ Takes a few seconds to provision new containers.
  ◦ Auto scales.

• **Modifiable**
  ◦ Can be modified to be used as just an IDE or a full-fledged tutoring system with feedback generation, etc.

• **Extensible**
  ◦ Multiple modules can be integrated with ease such as feedback tools, compilers, etc.
Quality Attributes

- Reliable
  - Database is clustered for high availability and fault-tolerance.
  - Redundancy for almost all nodes.

- Performance
  - Dynamic load balancing.
  - Static caching.
Concluding Remarks

- < 21 months old system, Still in experimentation phase
- Framework allows us to plug-and-play different components
  - Compiler: gcc, clang, python, prolog
  - Feedback: semantic feedback, compiler message rewriting, some ad-hoc scripts
  - Automated Problem Generation: Ad-hoc programs
  - Automated Testcase Generation: KLEE
  - Analytics
Future Work

- Lots of Data waiting to be processed
- Many HCI issues to be resolved
- User Survey (students, TAs, instructors)
- Feedback tools have 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)
