C-Pack of IPAs: A C90 Program Benchmark of Introductory Programming Assignments

Pedro Orvalho [1], Mikoláš Janota [2] and Vasco Manquinho [1]

[1]INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
[2]CIIRC, Czech Technical University in Prague, Czechia

APR 24, Lisbon, Portugal

Saturday 20th April, 2024

Motivation

- The increasing demand for programming education has given rise to Massive Open Online Courses (MOOCs) focused on introductory programming assignments (IPAs):

Motivation

- The increasing demand for programming education has given rise to Massive Open Online Courses (MOOCs) focused on introductory programming assignments (IPAs):
  - In 2018, MIT's MOOC, Introduction to CS, reached 1.2 M enrollments;

Motivation

- The increasing demand for programming education has given rise to Massive Open Online Courses (MOOCs) focused on introductory programming assignments (IPAs):
  - In 2018, MIT's MOOC, Introduction to CS, reached 1.2 M enrollments;
  - In 2020, Stanford's CS MOOC had more than 10 K students.

Motivation

- Providing personalized feedback to this large number of enrolled students, **requires substantial time and effort from the faculty**;

Motivation

- Providing personalized feedback to this large number of enrolled students, **requires substantial time and effort from the faculty**;
- Automated Program Repair (APR) has become crucial to provide automatic personalized feedback to each student;

Motivation

Pressing demand for a more extensive and varied selection of IPAs.

Motivation

Pressing demand for a more extensive and varied selection of IPAs.

- The number of publicly available benchmarks of IPAs is significantly small;

Motivation

Pressing demand for a more extensive and varied selection of IPAs.

- The number of publicly available benchmarks of IPAs is significantly small;
- Publicly available datasets of IPAs generally fall into two categories:

Motivation

Pressing demand for a more extensive and varied selection of IPAs.

- The number of publicly available benchmarks of IPAs is significantly small;
- Publicly available datasets of IPAs generally fall into two categories:
  - Small collection of different IPAs.

Motivation

Pressing demand for a more extensive and varied selection of IPAs.

- The number of publicly available benchmarks of IPAs is significantly small;
- Publicly available datasets of IPAs generally fall into two categories:
  - Small collection of different IPAs.
    - e.g., the IntroClass [Le Goues et al., 2015] includes only six distinct IPAs;

Motivation

Pressing demand for a more extensive and varied selection of IPAs.

- The number of publicly available benchmarks of IPAs is significantly small;
- Publicly available datasets of IPAs generally fall into two categories:
  - Small collection of different IPAs.
    - e.g., the IntroClass [Le Goues et al., 2015] includes only six distinct IPAs;
  - Lack a comprehensive range of correct implementations for each IPA.

Motivation

Pressing demand for a more extensive and varied selection of IPAs.

- The number of publicly available benchmarks of IPAs is significantly small;
- Publicly available datasets of IPAs generally fall into two categories:
  - Small collection of different IPAs.
    - e.g., the IntroClass [Le Goues et al., 2015] includes only six distinct IPAs;
  - Lack a comprehensive range of correct implementations for each IPA.
    - e.g., the number of correct programs per IPA is similar to the incorrect ones in the ITSP dataset [Yi et al., 2017].

C-Pack-IPAs

C-Pack-IPAs is a publicly available benchmark comprising student-program submissions for **25 distinct IPAs**.
Currently, C-Pack-IPAs contains :

C-Pack-IPAs

C-Pack-IPAs is a publicly available benchmark comprising student-program submissions for **25 distinct IPAs**.
Currently, C-Pack-IPAs contains :

- semantically correct, semantically incorrect, and syntactically incorrect programs;

C-Pack-IPAs

C-Pack-IPAs is a publicly available benchmark comprising student-program submissions for **25 distinct IPAs**.
Currently, C-Pack-IPAs contains :

- semantically correct, semantically incorrect, and syntactically incorrect programs;
- a description and a test suite for each IPA;

C-Pack-IPAs

C-Pack-IPAs is a publicly available benchmark comprising student-program submissions for **25 distinct IPAs**.
Currently, C-Pack-IPAs contains :

- semantically correct, semantically incorrect, and syntactically incorrect programs;
- a description and a test suite for each IPA;
- a reference implementation for each IPA;

C-Pack-IPAs

C-Pack-IPAs is a publicly available benchmark comprising student-program submissions for **25 distinct IPAs**.
Currently, C-Pack-IPAs contains :

- semantically correct, semantically incorrect, and syntactically incorrect programs;
- a description and a test suite for each IPA;
- a reference implementation for each IPA;
- programs submitted over three academic years from 102 different students.

C-Pack-IPAs

- C-Pack-IPAs is intended to help the evaluation of novel automated program repair frameworks, **addressing both semantic and syntactic aspects**;

C-Pack-IPAs

- C-Pack-IPAs is intended to help the evaluation of novel automated program repair frameworks, **addressing both semantic and syntactic aspects**;
- A subset of C-Pack-IPAs's semantically incorrect programs has been **manually fixed and annotated**, to incorporate several program features;

C-Pack-IPAs

Table: High-level Description of C-Pack-IPAs Benchmark.

Labs	#IPAs	#Correct Submissions	#Semantically Incorrect Submissions	#Syntactically Incorrect Submissions
Lab02	10	799	486	223
Lab03	7	351	699	106
Lab04	8	465	246	119
Total	**25**	**1615**	**1434**	**448**

- Lab 02: integers, floats, IO operations, if-statements, and simple loops.
- Lab 03: loops, nested loops, auxiliary functions, and chars.
- Lab 04: integer arrays and strings.

Annotated Programs

Lab02 - Ex02: Read two integers N, M and print the smallest of them in the first row and the largest in the second.

```
1  int main()
2  {
3    int n, m;
4    scanf("%d%d", &n, &m);
5
6    if (n > m)
7      printf("%d\n%d\n", n, m);
8    else
9      printf("%d\n%d\n", m, n);
10
11   return 0;
12 }
```

Annotated Programs

Lab02 - Ex02: Read two integers N, M and print the smallest of them in the first row and the largest in the second.

```
1  int main()
2  {
3    int n, m;
4    scanf("%d%d", &n, &m);
5
6    if (n > m) // should be n < m
7      printf("%d\n%d\n", n, m);
8    else
9      printf("%d\n%d\n", m, n);
10
11   return 0;
12 }
```

Example (Annotations)

- #Variables: 2
- #Passed Tests: 0
- #Failed Tests: 4
- Tests:
  - t_0: Wrong Answer;
  - t_1: Wrong Answer;
  - t_2: Wrong Answer;
  - t_3: Wrong Answer;

Annotated Programs

Lab02 - Ex02: Read two integers N, M and print the smallest of them in the first row and the largest in the second.

```
1  int main()
2  {
3    int n, m;
4    scanf("%d%d", &n, &m);
5
6    if (n > m) // should be n < m
7      printf("%d\n%d\n", n, m);
8    else
9      printf("%d\n%d\n", m, n);
10
11   return 0;
12 }
```

Example (Annotations)

- #Faults: 1
- Faults: ['n > m']
- Faulty Lines: [6]
- Fault Types: [Wrong Comparison Operator]
- Repair Actions: [Replace]
- Suggested Repairs: ['n < m']

Annotated Programs

Lab02 - Ex02: Read two integers N, M and print the smallest of them in the first row and the largest in the second.

```c
int main()
{
  int n, m;
  scanf("%d%d", &n, &m);

  if (n > m) // should be n < m
    printf("%d\n%d\n", n, m);
  else
    printf("%d\n%d\n", m, n);

  return 0;
}
```

Example (Annotations)

- Next Correct Submission: year-1/lab02/ex02/ex02-stu_006-sub_004

Experimental Results

Results

- To create a baseline for C-Pack-IPAs, we used Clara [Gulwani et al., 2018] and Verifix [Ahmed et al., 2022];

Results

- To create a baseline for C-Pack-IPAs, we used Clara [Gulwani et al., 2018] and Verifix [Ahmed et al., 2022];
- We focused on the set of semantically incorrect programs (1434 programs);

Results

- To create a baseline for C-Pack-IPAs, we used Clara [Gulwani et al., 2018] and Verifix [Ahmed et al., 2022];
- We focused on the set of semantically incorrect programs (1434 programs);
- All the experiments were conducted using:

Results

- To create a baseline for C-Pack-IPAs, we used Clara [Gulwani et al., 2018] and Verifix [Ahmed et al., 2022];
- We focused on the set of semantically incorrect programs (1434 programs);
- All the experiments were conducted using:
  - a memory limit of **32GB**;

Results

- To create a baseline for C-Pack-IPAs, we used Clara [Gulwani et al., 2018] and Verifix [Ahmed et al., 2022];
- We focused on the set of semantically incorrect programs (1434 programs);
- All the experiments were conducted using:
  - a memory limit of **32GB**;
  - a timeout of **600 seconds**.

Results

	# Programs Fixed			
Repair Method	**Lab 02**	**Lab 03**	**Lab 04**	**Total**
Verifix	93 (19.14%)	0 (0.0%)	0 (0.0%)	93 (6.49%)
Clara (No Clusters)	275 (56.58%)	11 (1.57%)	0 (0.0%)	286 (19.94%)
Clara (Clusters)	346 (71.19%)	168 (24.03%)	36 (14.63%)	328 (38.35%)

There is room for improvement!

	# Programs Fixed			
Repair Method	**Lab 02**	**Lab 03**	**Lab 04**	**Total**
Verifix	93 (19.14%)	0 (0.0%)	0 (0.0%)	93 (6.49%)
Clara (No Clusters)	275 (56.58%)	11 (1.57%)	0 (0.0%)	286 (19.94%)
Clara (Clusters)	346 (71.19%)	168 (24.03%)	36 (14.63%)	**328 (38.35%)**

C-Pack-IPAs is a **valuable resource for developing advanced repair tools**.

Takeaway Message

- C-Pack-IPAs **contains both semantically and syntactically incorrect programs** and diverse correct implementations for each IPA;

Takeaway Message

- C-Pack-IPAs **contains both semantically and syntactically incorrect programs** and diverse correct implementations for each IPA;
- A subset of C-Pack-IPAs's **semantically incorrect programs has been manually fixed and annotated**, to incorporate several program features;

Takeaway Message

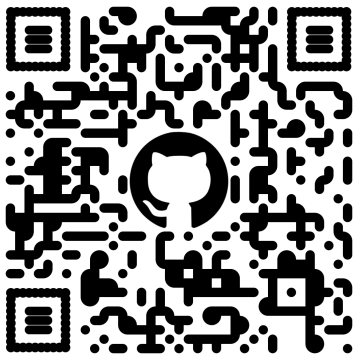
- C-Pack-IPAs **contains both semantically and syntactically incorrect programs** and diverse correct implementations for each IPA;
- A subset of C-Pack-IPAs's **semantically incorrect programs has been manually fixed and annotated**, to incorporate several program features;
- C-Pack-IPAs **is organized chronologically**, featuring submissions from 102 different students across three academic years;

Takeaway Message

- C-Pack-IPAs **contains both semantically and syntactically incorrect programs** and diverse correct implementations for each IPA;
- A subset of C-Pack-IPAs's **semantically incorrect programs has been manually fixed and annotated**, to incorporate several program features;
- C-Pack-IPAs **is organized chronologically**, featuring submissions from 102 different students across three academic years;
- Clara's 38.4% repair rate suggests **room for improvement** within the benchmark, which serves as a **valuable resource for developing advanced repair tools**.

C-Pack-IPAs

Thank you!



https://github.com/pmorvalho/C-Pack-IPAs

References


J. Yi and U. Z. Ahmed and A. Karkare and S. Hwei Tan and A. Roychoudhury (2017)
A feasibility study of using automated program repair for introductory programming assignments.
ESEC/FSE 2017.


Le Goues, C. and Holtschulte, N. and Smith, E. K and Brun, Y. and Devanbu, P. and Forrest, S. and Weimer, W. (2015)
The ManyBugs and IntroClass benchmarks for automated repair of C programs.
IEEE Transactions on Software Engineering (TSE) 2015.


Ahmed, U. Z and Fan, Z. and Yi, J. and Al-Bataineh, O. I and Roychoudhury, A. (2022)
Verifix: Verified repair of programming assignments.
TOSEM 22 12(3), 45 – 678.


Gulwani, Sumit and Radiček, Ivan and Zuleger, Florian (2018)
Automated clustering and program repair for introductory programming assignments.
PLDI 18 52(4), 465 – 480.


Orvalho, P. and Janota, M. and Manquinho, V. (2022)
C-Pack of IPAs: A C90 Program Benchmark of Introductory Programming Assignments.
arXiv:2206.08768.

Appendix

Lab 02: Types of Faults

Table: The number of faulty programs in each exercise of Lab02, with different types of program faults.

	Lab 02									
Fault Type	**Ex 01**	**Ex 02**	**Ex 03**	**Ex 04**	**Ex 05**	**Ex 06**	**Ex 07**	**Ex 08**	**Ex 09**	**Ex 10**
Incomplete Binary Operation	1			2						
Incorrect Data Type						1		3		
Incorrect Input	4	4	3			1				
Incorrect Output	63	18	11	14	3	3	3	6	17	3
Misplaced Expression						1				
Misplaced Loop Decrement						1				
Missing Expression						2	6			
Missing Instruction	1		2	8						
Missing Instructions				5						
Missing Loop Decrement						2				
Missing Loop Increment						1		1		
Missing Output	1									
Missing Variable					1	2	6			
Non-zero Return	1									
Presentation Error	44	26	22	27	2	1	6		3	2
Uninitialized Variable	11			12	1	3	1	5		
Variable Misuse	2	1	3	5	1		1		2	
Wrong Binary Operation				4		2	1			1
Wrong Comparison Operator	3	12		3				1		1
Wrong Exercise	8	2		2	1					
Wrong Expression				7	1	2	6	1	3	1
Wrong Initialization	1						1			
Wrong Instruction	5									
Wrong Literal			1		1	2		2	3	
Wrong Parameter								1		
Average Number of Faults	1.8	1.56	1.28	2.06	1.33	3.56	2.36	1.91	1.58	1.43

Repairing Lab 02

In Lab 02, the students learn how to program with integers, floats, IO operations (mainly `printf` and `scanf`), if-statements, and simple loops.

		Lab 02			
		Unsuccessful			
Repair Method	**% Fixed**	**% Structural Mismatch**	**% Unsupported Features**	**% Other Errors/Exceptions**	**% Timeouts (600s)**
Verifix	93 (19.14%)	92 (18.93%)	55 (11.32%)	246 (50.62%)	0 (0.0%)
Clara (No Clusters)	275 (56.58%)	210 (43.21%)	0 (0.0%)	1 (0.21%)	0 (0.0%)
Clara (Clusters)	346 (71.19%)	12 (2.47%)	0 (0.0%)	33 (6.79%)	95 (19.55%)

Repairing Lab 03

In Lab 03, the students learn how to program with loops, nested loops, auxiliary functions, and chars.

	Lab 03			
		Unsuccessful		
Repair Method	**% Fixed**	**% Structural Mismatch**	**% Unsupported Features**	**% Other Errors/Exceptions**
Verifix	0 (0.0%)	0 (0.0%)	699 (100.0%)	0 (0.0%)
Clara (No Clusters)	11 (1.57%)	511 (73.1%)	138 (19.74%)	39 (5.58%)
Clara (Clusters)	168 (24.03%)	65 (9.3%)	138 (19.74%)	328 (46.92%)

Repairing Lab 04

In Lab 04, the students learn how to program with integer arrays and strings.

		Lab 04		
		Unsuccessful		
Repair Method	**% Fixed**	**% Structural Mismatch**	**% Unsupported Features**	**% Other Errors/Exceptions**
Verifix	0 (0.0%)	6 (2.44%)	237 (96.34%)	3 (1.22%)
Clara (No Clusters)	0 (0.0%)	107 (43.5%)	138 (56.1%)	1 (0.41%)
Clara (Clusters)	36 (14.63%)	18 (7.32%)	138 (56.1%)	54 (21.95%)