new programming language user study

Welcome to the survey

ID: 10

Hello there!

Great to have you here and thank you for your willingness to participate in our online user study.

Introduction to the topic

In this online study we will ask you to assess the usability of a new programming language for intermittently-powered embedded devices. These tiny devices are powered from ambient energy sources like solar energy, WiFi transmissions, or vibrations. These devices store their energy in small capacitors or very small batteries. Small energy storage make them super-small in size (which is great!). But this also causes very frequent power outages due to the intermittent nature of the energy sources, the constrained size of the storage, and the expensive communication and compute tasks. Some examples of intermittently-powered devices include an ambient backscatter node, a micro-satelite, or a intermittent robot.

To guarantee the progress of computation from one power outage to another (which sometimes might be in the order of dozens per second) programmers need to know when to store the intermediate result of the computation and how much data must be saved. Progress of a program is stored in a non-volatile memory. But care must be taken when storing data, as corruption can occur (a variable might be incremented more than required due to being saved multiple times at each power outage). The role of a programming language is to protect the application and the programmer from such errors through explicit semantics, while causing minimum overhead to the system (fewer protection routines means less overhead but higher chance of being interrupted by outage) and to the programmer (more protection usually means more code to write and handle).

Goal of this survey

There are of course many ways to design a programming language for intermittently-powered devices. Most of them trade-off ease of use with efficiency of the code execution. The goal of this study is to see how much time a programmer spends on writing programs using two main classes of languages for intermittently-powered devices. We will assess this by measuring time spent searching for bugs in simple C language-like programs written in two programming models belonging to each of these language classes. One of them has been designed in the past month by this research team.

In the next page we will provide you a quick tutorial on these two classes of programming languages.

What we ask you to do

Please carefully look into the code and click next only when you are sure you have found the bug. Do not rush with the answer. And don't worry - once you complete all questions we will share the answers with you.

What preparation is needed

A basic understanding of the C programming language and its syntax will be sufficient to complete this survey.

Time to complete

The survey can be done at your own pace; however, we do not expect it to take more than 30 minutes of your time.

Hidden Value: New Hidden Value

Value: Populates with the **length of time** since the survey taker started the current page

---

Crash course on programming languages for Intermittently-powered devices

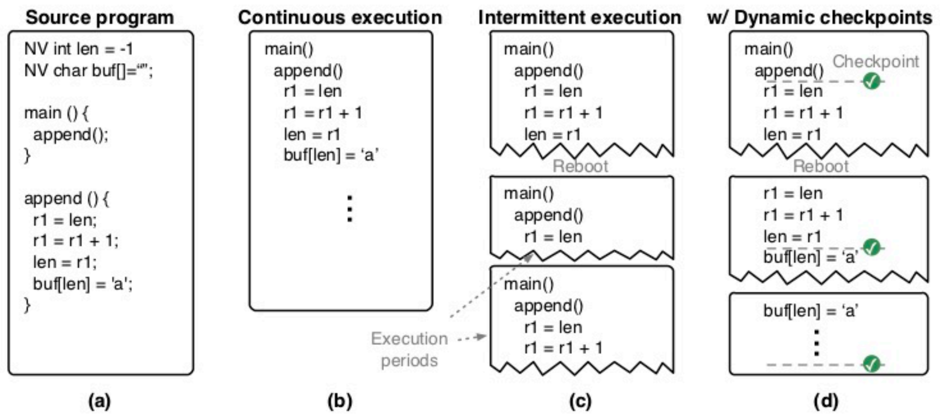
Let us first give a quick description of two main classes of programming languages for intermittently-powered devices

Ready?

ID: 12

Checkpointing-based Language

In this approach a compiler or a programmer checkpoints to a non-volatile memory a complete state of a program (stack, registers, etc.) or some state of the program (selected variables) at predefined positions of the C program. Examples of existing programming languages belonging to this class include *DINO* and *Mementos*.



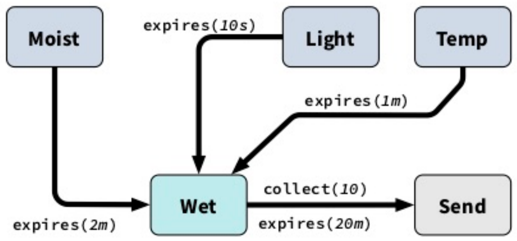
Source: *A simpler, safer programming and execution model for intermittent systems*, in Proc. PLDI'15

Checkpointing works like the above figure (d), where at certain instances state is preserved. Our proposal is a new checkpointing-based language which avoids the requirement by the programmer of picking checkpoint locations. The programmer only needs to add one header file in an existing C code and initialize checkpoint routine right after invocation of the *main()* function (for instance *checkpoint_initialization()*).

ID: 13

Task-based Language

In this approach a programmer needs to convert a C program into its equivalent, based on tasks. A task is a chain of functions that connect with each other by passing its output to the input of a new task. Examples of existing programming languages belonging to this class include *InK and Alpaca*, or Mayfly (shown below). The below program gathers moisture, light, and temperature readings, calculates a wetness value, and sends that value in a buffer to the base station for greenhouse monitoring.



Source: *Timely Execution on Intermittently Powered Batteryless Sensors*, in Proc. SenSys'17

Looking in more detail, in the task based language such as InK, a program needs to be initialized first, by including a header file and then by adding the following functions:

ENTRY_TASK(name_of_first_task); //first task
TASK(name_of_second_task); //second task
...
TASK(name_of_Xth_task); //Xth task

Then the programmer needs to define all variables as global ones by invoking

SHARED(type1 variable 1, type2 variable 2, ...);

Then, the programmer needs to define each task, in the following fashion:

```
TASK(name_of_a_task)
{
  SET(y,3); //initialize y to y = 3
  SET(x, GET(y)); //assign new value to x by using protected value y
  return name_of_a_new_task_to_which_this_task_connects_to;
}
```

That's about it!

ID: 35

How to Answer the Questions

In the flowing questions you will be presented with two implementations of three different types of algorithms for a total of 6 questions. The two implementations consist of a checkpoint based language (language 1) and a task based language (language 2).

To make life easier all questions only contain one line that is buggy! Your task is to first state the line containing the bug, and then to correct that line.

Please (really!) do not rush and expend some effort to find the bug.

OK! We are now ready to take the survey!

Hidden Value: Time spend on explanation

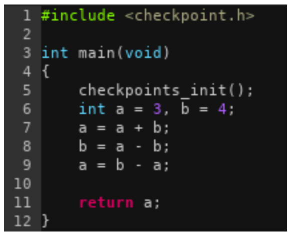
Value: Populates with the **length of time** since the survey taker started the current page

Find a bug for the swapping algorithm using checkpoints (language 1)

ID: 4

1) This programs swaps two values *without* the use of a third temporary variable, and is implemented using checkpoints (language 1). The swapping is implemented by three operations: $X = X + Y$ then $Y = X - Y$ and finally $X = X - Y$. That's it!

The code depicted bellow contains one bug related to the swapping of the variables.



```c
 1 #include <checkpoint.h>
 2 
 3 int main(void)
 4 {
 5     checkpoints_init();
 6     int a = 3, b = 4;
 7     a = a + b;
 8     b = a - b;
 9     a = b - a;
10 
11     return a;
12 }
```

Select the line containing this bug.*

Line 1

Line 2

Line 3

Line 4

Line 5

Line 6

Line 7

Line 8

Line 9

Line 10

Line 11

Line 12

ID: 6

2) Write a correct statement for the line that you think contains a bug. Please use C syntax only (no sentences describing what is the error).*

Hidden Value: Time spent swap checkpoint

Value: Populates with the **length of time** since the survey taker started the current page

Find a bug for the swapping algorithm using tasks (language 2)

ID: 39

3) A recap of the task based language

In the task based language, a program needs to be initialized first, by including a header file and then by adding the following functions:

ENTRY_TASK(name_of_first_task); //first task
TASK(name_of_second_task); //second task
...
TASK(name_of_Xth_task); //Xth task

Then the programmer needs to define all variables as global ones by invoking

SHARED(type1 variable 1, type2 variable 2, ...);

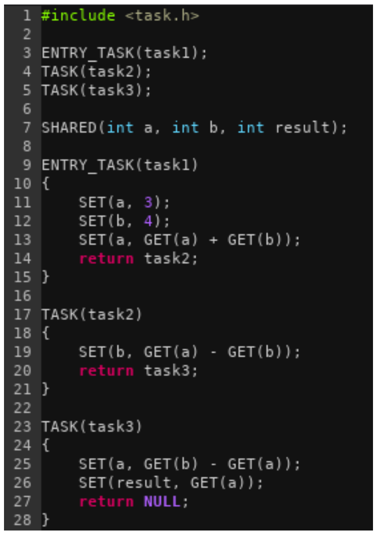
Then, the programmer needs to define each task, in the following fashion:

```
TASK(name_of_a_task)
{
  SET(y,3); //initialize y to y = 3
  SET(x, GET(y)); //assign new value to x by using protected value y
  return name_of_a_new_task_to_which_this_task_connects_to;
}
```

The Problem

This programs swaps two values without the use of a third temporary variable, and is implemented using tasks (language 2). The swapping is implemented by three operations: X = X + Y then Y = X - Y and finally X = X - Y. That's it!

The code depicted bellow contains one bug related to the swapping of the variables.



```
#include <task.h>

ENTRY_TASK(task1);
TASK(task2);
TASK(task3);

SHARED(int a, int b, int result);

ENTRY_TASK(task1)
{
    SET(a, 3);
    SET(b, 4);
    SET(a, GET(a) + GET(b));
    return task2;
}

TASK(task2)
{
    SET(b, GET(a) - GET(b));
    return task3;
}

TASK(task3)
{
    SET(a, GET(b) - GET(a));
    SET(result, GET(a));
    return NULL;
}
```

Which line contains an error?
 *****

Line 1

Line 2

Line 3

Line 4

Line 5

Line 6

Line 7

Line 8

Line 9

Line 10

Line 11

Line 12

Line 13

Line 14

Line 15

Line 16

Line 17

Line 18

Line 19

Line 20

Line 21

Line 22

Line 23

Line 24

Line 25

Line 26

Line 27

Line 28

ID: 40

4) Write a correct statement for the line that you think contains a bug. Please use C syntax only (no sentences describing what is the error).*

Hidden Value: Time spent swap task

Value: Populates with the **length of time** since the survey taker started the current page

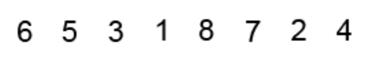
---

Find a bug in Bubble sort using checkpoints (language 1)

ID: 98

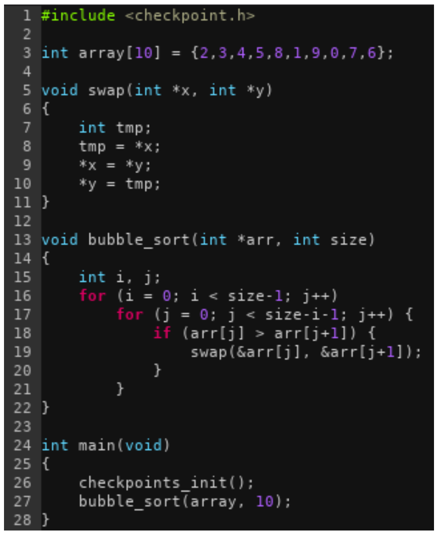
5) This program performs a Bubble sort and makes use of checkpoints (language 1).
The goal of the program is to sort an array of integers.

Quick recap on Bubble sort



Source: https://en.wikipedia.org/wiki/Bubble_sort

The code depicted bellow contains one bug related to the operation of the Bubble sort algorithm.



```
 1 #include <checkpoint.h>
 2
 3 int array[10] = {2,3,4,5,8,1,9,0,7,6};
 4
 5 void swap(int *x, int *y)
 6 {
 7     int tmp;
 8     tmp = *x;
 9     *x = *y;
10     *y = tmp;
11 }
12
13 void bubble_sort(int *arr, int size)
14 {
15     int i, j;
16     for (i = 0; i < size-1; j++)
17         for (j = 0; j < size-i-1; j++) {
18             if (arr[j] > arr[j+1]) {
19                 swap(&arr[j], &arr[j+1]);
20             }
21         }
22 }
23
24 int main(void)
25 {
26     checkpoints_init();
27     bubble_sort(array, 10);
28 }
```

Select the line containing this bug.*

Line 1

Line 2

Line 3

Line 4

Line 5

Line 6

Line 7

Line 8

Line 9

Line 10

Line 11

Line 12

Line 13

Line 14

Line 15

Line 16

Line 17

Line 18

Line 19

Line 20

Line 21

Line 22

Line 23

Line 24

Line 25

Line 26

Line 27

Line 28

ID: 99

6) Write a correct statement for the line that you think contains the bug. Please use C syntax only (no sentences describing what is the error).*

Hidden Value: Time spent bubble sort checkpoint

Value: Populates with the **length of time** since the survey taker started the current page

Find a bug in Bubble sort using tasks (language 2)

ID: 104

7) A recap of the task based language

In the task based language, a program needs to be initialized first, by including a header file and then by adding the following functions:

ENTRY_TASK(name_of_first_task); //first task
TASK(name_of_second_task); //second task
...
TASK(name_of_Xth_task); //Xth task

Then the programmer needs to define all variables as global ones by invoking

SHARED(type1 variable 1, type2 variable 2, ...);

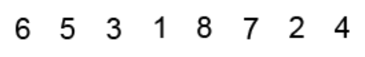
Then, the programmer needs to define each task, in the following fashion:

```
TASK(name_of_a_task)
{
  SET(y,3); //initialize y to y = 3
  SET(x, GET(y)); //assign new value to x by using protected value y
  return name_of_a_new_task_to_which_this_task_connects_to;
}
```

The Problem

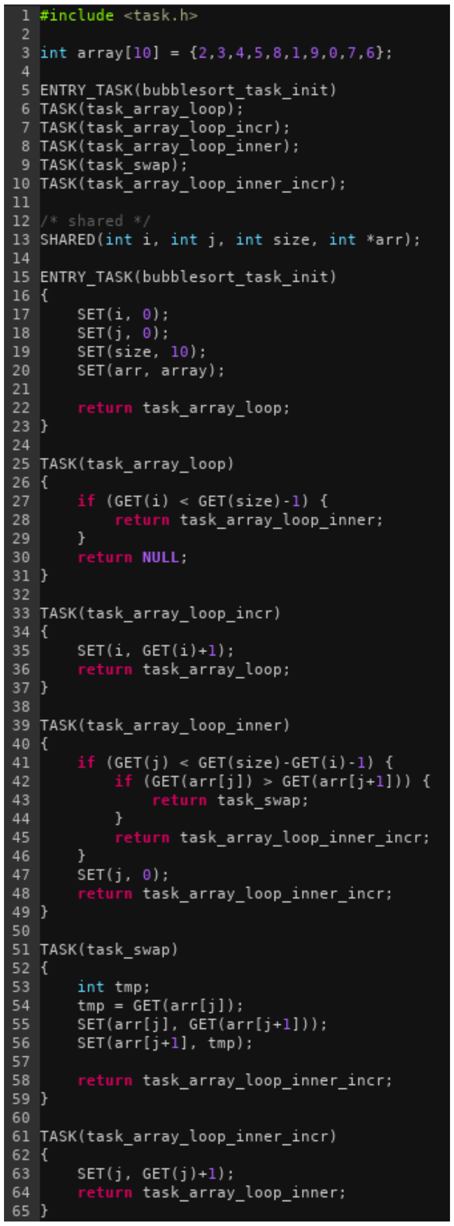
Copy of This program performs a Bubble sort and makes use of tasks (language 2).
The goal of the program is to sort an array of integers.

Quick recap on Bubble sort



Source: https://en.wikipedia.org/wiki/Bubble_sort

The code depicted bellow contains one bug related to the operation of the Bubble sort algorithm.



```
 1 #include <task.h>
 2 
 3 int array[10] = {2,3,4,5,8,1,9,0,7,6};
 4 
 5 ENTRY_TASK(bubblesort_task_init)
 6 TASK(task_array_loop);
 7 TASK(task_array_loop_incr);
 8 TASK(task_array_loop_inner);
 9 TASK(task_swap);
10 TASK(task_array_loop_inner_incr);
11 
12 /* shared */
13 SHARED(int i, int j, int size, int *arr);
14 
15 ENTRY_TASK(bubblesort_task_init)
16 {
17     SET(i, 0);
18     SET(j, 0);
19     SET(size, 10);
20     SET(arr, array);
21 
22     return task_array_loop;
23 }
24 
25 TASK(task_array_loop)
26 {
27     if (GET(i) < GET(size)-1) {
28         return task_array_loop_inner;
29     }
30     return NULL;
31 }
32 
33 TASK(task_array_loop_incr)
34 {
35     SET(i, GET(i)+1);
36     return task_array_loop;
37 }
38 
39 TASK(task_array_loop_inner)
40 {
41     if (GET(j) < GET(size)-GET(i)-1) {
42         if (GET(arr[j]) > GET(arr[j+1])) {
43             return task_swap;
44         }
45         return task_array_loop_inner_incr;
46     }
47     SET(j, 0);
48     return task_array_loop_inner_incr;
49 }
50 
51 TASK(task_swap)
52 {
53     int tmp;
54     tmp = GET(arr[j]);
55     SET(arr[j], GET(arr[j+1]));
56     SET(arr[j+1], tmp);
57 
58     return task_array_loop_inner_incr;
59 }
60 
61 TASK(task_array_loop_inner_incr)
62 {
63     SET(j, GET(j)+1);
64     return task_array_loop_inner;
65 }
```

Select the line containing this bug.*

Line 1

Line 2

Line 3

Line 4

Line 5

Line 6

Line 7

Line 8

Line 9

Line 10

Line 11

Line 12

Line 13

Line 14

Line 15

Line 16

Line 17

Line 18

Line 19

Line 20

Line 21

Line 22

Line 23

Line 24

Line 25

Line 26

Line 27

Line 28

Line 29

Line 30

Line 31

Line 32

Line 33

Line 34

Line 35

Line 36

Line 37

Line 38

Line 39

Line 40

Line 41

Line 42

Line 43

Line 44

Line 45

Line 46

Line 47

Line 48

Line 49

Line 50

Line 51

Line 52

Line 53

Line 54

Line 55

Line 56

Line 57

Line 58

Line 59

Line 60

Line 61

Line 62

Line 63

Line 64

Line 65

ID: 102

8) Write a correct statement for the line that you think contains the bug. Please use C syntax only (no sentences describing what is the error).*

Hidden Value: Time spent bubble sort task

Value: Populates with the **length of time** since the survey taker started the current page

---

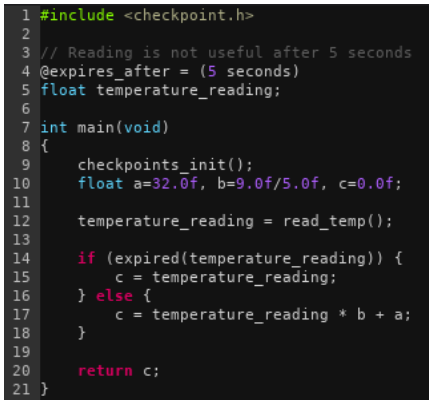
Find a bug in the time dependent program using checkpoints (language 1)

ID: 88

9) This is a program that makes use of *time* to make decisions. Keeping track of time across power failures is hard, because the time values resets to zero after every power failure. Modern runtimes handle this by using *capacitive timekeepers*. For the below code, assume the runtime handles timekeeping.

In this program, we want to read the temperature, and convert it to Fahrenheit if it is not expired. Otherwise we set the temperature to 32.

The code depicted bellow contains one bug compared to the description above.



```
 1 #include <checkpoint.h>
 2 
 3 // Reading is not useful after 5 seconds
 4 @expires_after = (5 seconds)
 5 float temperature_reading;
 6 
 7 int main(void)
 8 {
 9     checkpoints_init();
10     float a=32.0f, b=9.0f/5.0f, c=0.0f;
11 
12     temperature_reading = read_temp();
13 
14     if (expired(temperature_reading)) {
15         c = temperature_reading;
16     } else {
17         c = temperature_reading * b + a;
18     }
19 
20     return c;
21 }
```

Select the line containing this bug.*

Line 1

Line 2

Line 3

Line 4

Line 5

Line 6

Line 7

Line 8

Line 9

Line 10

Line 11

Line 12

Line 13

Line 14

Line 15

Line 16

Line 17

Line 18

Line 19

Line 20

Line 21

ID: 89

10) Write a correct statement for the line that you think contains a bug. Please use C syntax only (no sentences describing what is the error).*

Hidden Value: Time spent time based checkpoint

Value: Populates with the **length of time** since the survey taker started the current page

Find a bug in the time dependent program using tasks (language 2)

ID: 91

11) A recap of the task based language

In the task based language, a program needs to be initialized first, by including a header file and then by adding the following functions:

```
ENTRY_TASK(name_of_first_task); //first task
TASK(name_of_second_task); //second task
...
TASK(name_of_Xth_task); //Xth task
```

Then the programmer needs to define all variables as global ones by invoking

```
SHARED(type1 variable 1, type2 variable 2, ...);
```

Then, the programmer needs to define each task, in the following fashion:

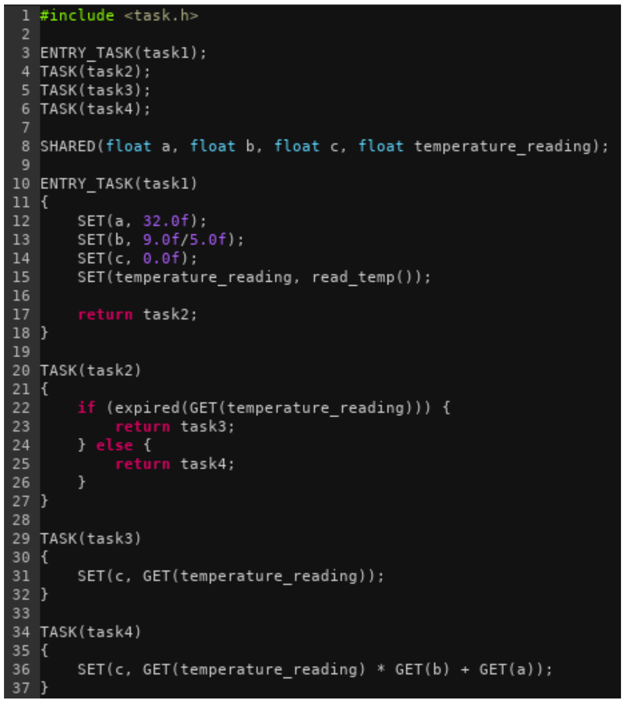
```
TASK(name_of_a_task)
{
  SET(y,3); //initialize y to y = 3
  SET(x, GET(y)); //assign new value to x by using protected value y
  return name_of_a_new_task_to_which_this_task_connects_to;
}
```

The Problem

This is a program that makes use of *time* to make decisions. Keeping track of time across power failures is hard, because the time values resets to zero after every power failure. Modern runtimes handle this by using *capacitive timekeepers*. For the below code, assume the runtime handles timekeeping.

In this program, we want to read the temperature, and convert it to Fahrenheit if it is not expired. Otherwise we set the temperature to 32.

The code depicted bellow contains one bug according to the description above.



```
 1 #include <task.h>
 2 
 3 ENTRY_TASK(task1);
 4 TASK(task2);
 5 TASK(task3);
 6 TASK(task4);
 7 
 8 SHARED(float a, float b, float c, float temperature_reading);
 9 
10 ENTRY_TASK(task1)
11 {
12     SET(a, 32.0f);
13     SET(b, 9.0f/5.0f);
14     SET(c, 0.0f);
15     SET(temperature_reading, read_temp());
16 
17     return task2;
18 }
19 
20 TASK(task2)
21 {
22     if (expired(GET(temperature_reading))) {
23         return task3;
24     } else {
25         return task4;
26     }
27 }
28 
29 TASK(task3)
30 {
31     SET(c, GET(temperature_reading));
32 }
33 
34 TASK(task4)
35 {
36     SET(c, GET(temperature_reading) * GET(b) + GET(a));
37 }
```

Select the line containing this bug.
 *

Line 1

Line 2

Line 3

Line 4

Line 5

Line 6

Line 7

Line 8

Line 9

Line 10

Line 11

Line 12

Line 13

Line 14

Line 15

Line 16

Line 17

Line 18

Line 19

Line 20

Line 21

Line 22

Line 23

Line 24

Line 25

Line 26

Line 27

Line 28

Line 29

Line 30

Line 31

Line 32

Line 33

Line 34

Line 35

Line 36

Line 37

ID: 92

12) Write a correct statement for the line that you think contains a bug. Please use C syntax only (no sentences describing what is the error).*

Hidden Value: Time spent time based task

Value: Populates with the **length of time** since the survey taker started the current page

Subjective assessment of language 1 and language 2

How do you view these two languages?

Here, we would like to ask you questions about your subjective view of *language 1* and *language 2*.

ID: 42

13) Knowing Language 1 and Language 2, I think programming with this language would be difficult.*

	Strongly disagree	**Disagree**	**Neither agree or disagree**	**Agree**	**Strongly agree**
Language 1 (checkpoints)					
Language 2 (tasks)					

ID: 45

14) Knowing Language 1 and Language 2, I think programming with this language would be easy.*

	Strongly disagree	**Disagree**	**Neither agree or disagree**	**Agree**	**Strongly agree**
Language 1 (checkpoints)					
Language 2 (tasks)					

ID: 48

15) Knowing Language 1 and Language 2, I think the programming model is intuitive.*

	Strongly disagree	**Disagree**	**Neither agree or disagree**	**Agree**	**Strongly agree**
Language 1 (checkpoints)					
Language 2 (tasks)					

ID: 51

16) Knowing Language 1 and Language 2, I think the programming model is complete.*

	Strongly disagree	**Disagree**	**Neither agree or disagree**	**Agree**	**Strongly agree**
Language 1 (checkpoints)					
Language 2 (tasks)					

ID: 54

17) Knowing Language 1 and Language 2, I think this programming model is flexible.*

	Strongly disagree	**Disagree**	**Neither agree or disagree**	**Agree**	**Strongly agree**

Language 1 (checkpoints)					
Language 2 (tasks)					

ID: 94

18) Knowing Language 1 and Language 2, I think this programming model is concise.*

	Strongly disagree	**Disagree**	**Neither agree or disagree**	**Agree**	**Strongly agree**
Language 1 (checkpoints)					
Language 2 (tasks)					

ID: 57

19) Knowing Language 1 and Language 2, I think this programming model could be used for a variety of applications, like greenhouse monitoring, activity tracking, or signal processing.*

	Strongly disagree	**Disagree**	**Neither agree or disagree**	**Agree**	**Strongly agree**
Language 1 (checkpoints)					
Language 2 (tasks)					

ID: 61

20) Knowing Language 1 and Language 2, I think the programming model; including syntax, keywords, and control flow, is easy to understand.*

	Strongly disagree	**Disagree**	**Neither agree or disagree**	**Agree**	**Strongly agree**
Language 1 (checkpoints)					
Language 2 (tasks)					

ID: 64

21) Knowing Language 1 and Language 2, If I had to implement the program that implements *sense() -> compute() -> transmit()* from scratch (in this programming model), I don't think it would take me very long.*

	Strongly disagree	**Disagree**	**Neither agree or disagree**	**Agree**	**Strongly agree**
Language 1 (checkpoints)					
Language 2 (tasks)					

ID: 70

22) Knowing Language 1 and Language 2, If I had the choice, I would use this programming model for my own projects in Intermittently Powered Computing.*

	Strongly disagree	Disagree	Neither agree or disagree	Agree	Strongly agree
Language 1 (checkpoints)					
Language 2 (tasks)					

ID: 73

23) Knowing Language 1 and Language 2, If I had to teach a new student how to program Intermittently Powered Systems, I would use this language.*

	Strongly disagree	Disagree	Neither agree or disagree	Agree	Strongly agree
Language 1 (checkpoints)					
Language 2 (tasks)					

ID: 76

24) Knowing Language 1 and Language 2, I think this programming model is easier to grasp than Python, Ruby, or JavaScript.*

	Strongly disagree	Disagree	Neither agree or disagree	Agree	Strongly agree

Language 1 (checkpoints)					
Language 2 (tasks)					

ID: 79

25) Knowing Language 1 and Language 2, I think this programming model is easier to grasp than JAVA / C# / C++.*

	Strongly disagree	**Disagree**	**Neither agree or disagree**	**Agree**	**Strongly agree**
Language 1 (checkpoints)					
Language 2 (tasks)					

ID: 82

26) Knowing Language 1 and Language 2, I think this programming model is easier to grasp than ANSI C.*

	Strongly disagree	**Disagree**	**Neither agree or disagree**	**Agree**	**Strongly agree**
Language 1 (checkpoints)					
Language 2 (tasks)					

ID: 85

27) Knowing Language 1 and Language 2, It was easy to convey my intentions with this programming model.*

	Strongly disagree	**Disagree**	**Neither agree or disagree**	**Agree**	**Strongly agree**
Language 1 (checkpoints)					
Language 2 (tasks)					

Hidden Value: Time spend on difficulty

Value: Populates with the **length of time** since the survey taker started the current page

---

Information about yourself

Finally, we would like to know a bit about you.

We want to know how do you assess your *coding skills* and your knowledge of *topics related to this survey*. Also, we would like to know just a little bit regarding your demographics data.

ID: 15

28) How many years did you spend on any of these topics:*

	none	**1 year**	**2 years**	**3 years**	**4 years**	**5 years**	**more than 5 years**
Formal electrical engineering education (university-level)							
Formal computer science education (university-level)							
Programming language experience							
Practical embedded systems design							

ID: 19

29) Please self-assess yourself on the following question: "Compared to others with similar background, age and education my knowledge of [XYZ] is", where [XYZ] is*

	Below average	**Slightly below average**	**Average**	**Slightly above average**	**Above average**
C programming language					

Programming languages other than C					
Electronics					
Energy harvesting technologies					
Embedded systems					
Embedded systems powered by energy harvesting technologies					
Embedded systems powered intermittently					

ID: 29

30) My programming language of choice is

C

C++

Java

JavaScript

Python

Shell Script

Scala

PHP

Ruby

Rust

Assembly

Other

ID: 27

31) I am a"*

Man

Woman

Validation: Min = 18 Max = 110 Must be numeric Whole numbers only Positive numbers only Max character count = 3

ID: 28

32) My age is*

Hidden Value: Time spend on skills

Value: Populates with the **length of time** since the survey taker started the current page

---

Where are the bugs: the answers

You probably want to know the answers to the question where is the bug? Here they are!

- **For the variable swapping:**
  - **For checkpoints (language 1): bug at line 9 (it should be a = a - b;)**
  - **For tasks (language 2): bug at line 25 (it should be SET(a, GET(a) - GET(b));)**
- **For the Bubble sort:**
  - **For checkpoints (language 1): bug at line 16 (it should be i++)**

    - **For tasks (language 2): bug at line 48 (it should be return array_loop_incr;)**
- **For the time based variables:**
    - **For checkpoints (language 1): bug at line 15 (it should be c = a;)**
    - **For tasks (language 2): bug at line 31 (it should be SET(c, GET(a));)**

Hidden Value: Confirmation code

Value: [survey("counter"), safer="true", startat="100000"]

Thank You!

ID: 1

Thank you for taking our survey! Your response is very important to us.

Feel free to contact us for the survey results and pass around the link to the survey to your friends. We want to have the most diverse group of people possible! The link is here:

And here is the confirmation code: [question('value'), id='106']
