High Performance Rust MSc Questionnaire

Jim Walker

Hi! I'm Jim Walker, one of the MSc students at EPCC. My dissertation aims to examine if the new programming language Rust, is suitable for HPC. To that end, I have written this questionnaire, to asses how easy it is for HPC programmers such as yourself to understand it.

The questionnaire is simple. There are five questions, which present you with a fragment of rust code. Please describe what each of these functions does to the best of your abilities

Question 1

Select one correct answer.

```
let mut v1 = vec![2,8];
let v2 = vec![2;8];
```

- ☐ Assigns label v1 to a mutable vector of elements 2 and 8. Assigns label v2 to a vector with 8 elements of value 2.
- ☐ Assigns label v1 to a mutated vector with elements 2 and 8. Assigns label v2 to a vector of 2 elements, both with value 8.
- ☐ Assigns label v1 to a mutatable vector with elements 2 and 8. Assigns label v2 to a vector of 2 elements, both with value 8.
- ☐ Assigns label v1 to a mutable vector of 8 elements of value 2. Assigns label v2 to a vector with elements 2 and 8.

Question 2

Select one correct answer.

In this question, please assume that v is a vector.

```
v.iter().fold(1, |acc, x| acc * x);
```

- ☐ An iterator is created over the vector v, which calls the anonymous/lambda function on each element of the vector.
- ☐ Every element of v is multiplied together
- ☐ thing3
- ☐ thing4

Question 3

Select one correct answer.

What is printed?

```
let mut stack = Vec::new();

stack.push(1);
stack.push(2);
stack.push(3);

while let Some(top) = stack.pop() {
  print!("{} ", top);
}
```

☐ Some(3) Some(2) Some(1)

☐ 3 2 1 None None None...

☐ 3 2 1

☐ Some(3) Some(2) Some(1) None None None...

Question 4

Select one correct answer.

What is a set to?

```
let a: Vec<i32> = (1..).step_by(3)
                       .take(3)
                       .map(|x| x * 2)
                       .collect();
```

☐ [2, 4, 6]

☐ Syntax error

☐ [4, 10, 16]

☐ [2. 8, 14]

Question 5

Select one correct answer.

```
pub fn foo(&mut self)->T {

self.a.par_chunks(self.chunk_size)
    .zip(self.b.par_chunks(self.chunk_size))
    .map(|(a,b)| a.iter()
                  .zip(b.iter())
                  .fold(0, | acc, ele| acc + *ele.0 * *ele.1)
```

```
            )
        .sum()
}
```

- ☐ Sum reduction
- ☐ Dot Product
- ☐ Transpose?
- ☐ thing4

Question 6

I have ____ years of experience in HPC / Scientific Programming