

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 assess 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

In Rust, `vec!` is used to create a vector. All variables in Rust are immutable by default. If this program compiles and runs, what does it print?

```
let v = vec![2,3];
v.push(3);
println!("{:?}", v);
```

- ☐ [2,3,2]
- ☐ [2,2,2,3]
- ☐ The program does not compile.
- ☐ The program compiles, but crashes when it tries to push 3 to v.

Question 2

Idomatic Rust code often uses patterns associated with functional languages. Given an immutable vector, `v`, please select what the line of code below does.

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

- ☐ Every element of `v` is set to 1, and the result is stored in `a`.
- ☐ Every element of `v` is multiplied together and the result is stored in `a`.
- ☐ Every element of `v` is multiplied by 1 and the result is stored in `a`.
- ☐ The program does not compile.

Question 3

A vector's push method return an optional value, or none. What does this fragment of code print?

```
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

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, doesn't compile.
- ☐ [4, 10, 16]
- ☐ [2, 8, 14]

Question 5

In this question, a and b are both vectors of the same length. The method **par_chunks** returns a parallel iterator over at most **chunk_size** elements at a time. What does this fragment of code do?

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

- ☐ Sum reduction
- ☐ Dot Product
- ☐ Transpose
- ☐ A single iteration of a one dimensional relaxation.

Question 6

What does the function `foo` do?

```
fn foo(m: i32, n: i32) -> i32 {
    if m == 0 {
        n.abs()
    } else {
        foo(n % m, m)
        test()
    }
}
```

- ☐ It finds the greatest common divisor of m and n
- ☐ It doesn't compile.
- ☐ It finds the closest prime number to n
- ☐ It calls itself infinitely.

Question 7

The Rust compiler's borrow checker makes sure that values are mutably borrowed if they are altered from a different function than the one they were created in. What does this program print?

```
fn plus_one(x: &mut i32){
    *x += 1;
}
fn main(){
    let x = 64;
    plus_one(&mut x);
    println!("{}", x+1);
}
```

- ☐ 65
- ☐ Undefined.
- ☐ It doesn't compile.
- ☐ 66.

Question 8

I have ____ years of experience in HPC / Scientific Programming