./old/capy.jpg

CMSC330 Fall 2024 Quiz 4 Solutions

Problem 1: Basics

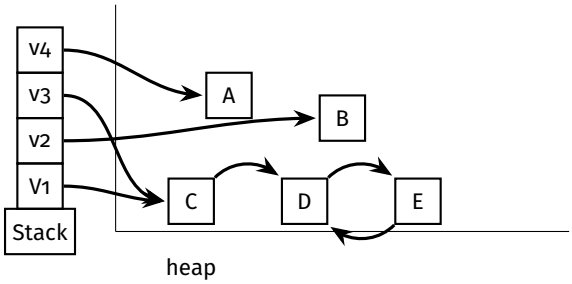
[Total 4 pts]

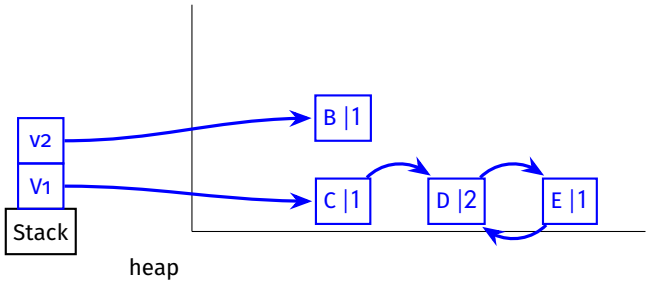
	True	False
Using only the reference counting garbage collection technique will result in certain data structures not being freed	T	F
Rust's Type System prevents use after free (ignoring the **unsafe** keyword and the RC module)	T	F
A reference's lifetime is part of its type	T	F
A conservative Garbage Collector will not free data if it is unsure if the data is still in use	T	F
Mark and sweep solves the problem of fragmentation in garbage collection	T	F
Rust's Type System allows use after free (ignoring the **unsafe** keyword and the RC module)	T	F
A variable's lifetime and their scope are the same thing.	T	F
Modern garbage collectors use one type of garbage collection algorithm exclusively.	T	F
Using reference counting garbage collection technique ensures that all data structures are freed	T	F
In Rust, values are freed once their lifetime ends.	T	F
A reference's lifetime is not a part of its type.	T	F
Mark and Sweep garbage collection technique halves usable memory.	T	F
Reference counting garbage collection technique ensures cyclic data structures gets freed	T	F
A piece of data can have multiple immutable references to it.	T	F
Modern Garbage Collectors use a mix of different garbage collection techniques.	T	F

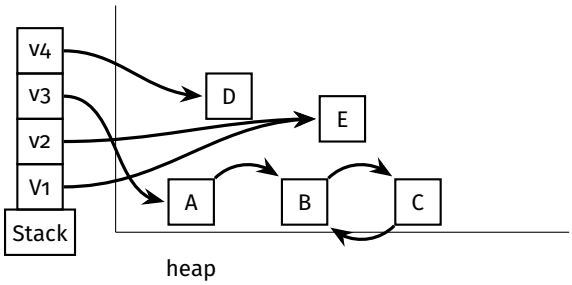
Problem 2: Garbage Collection

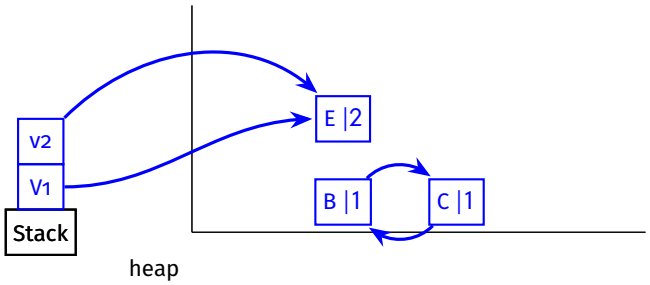
[Total 8 pts]

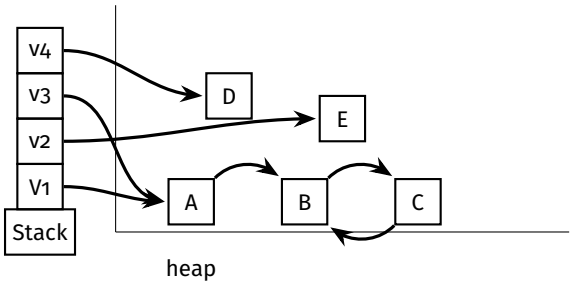
Suppose that *v*4 and *v*3 are popped off the stack. Using the reference counting technique, draw the stack and heap after this occurs.

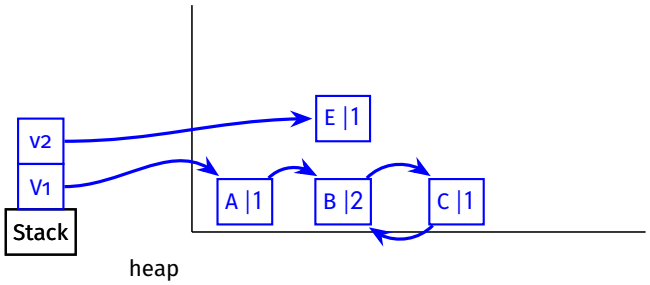


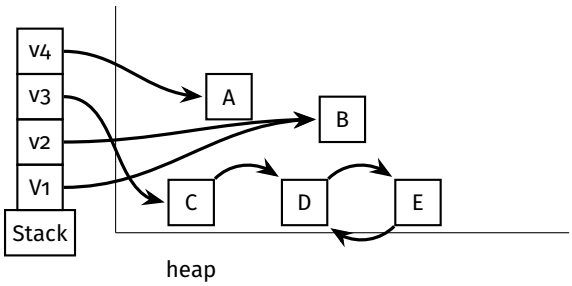


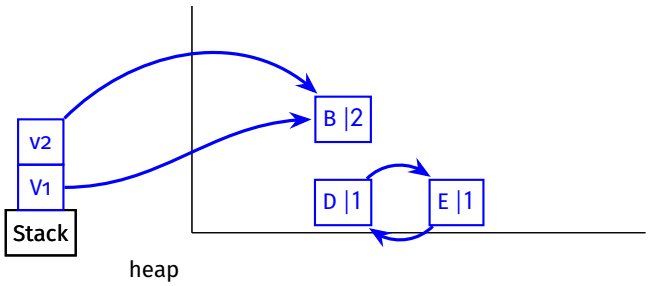




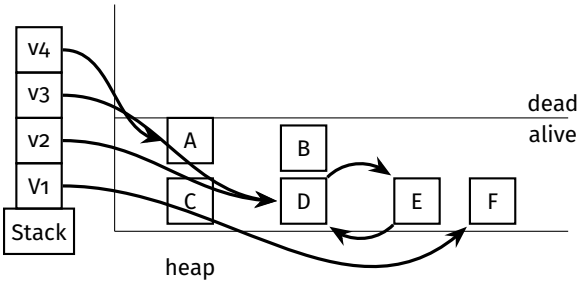


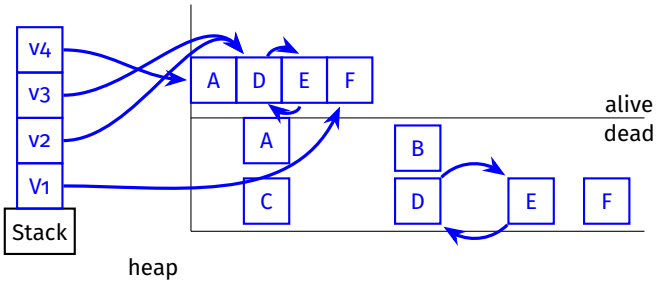


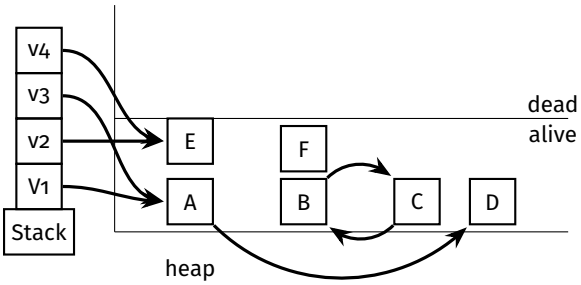


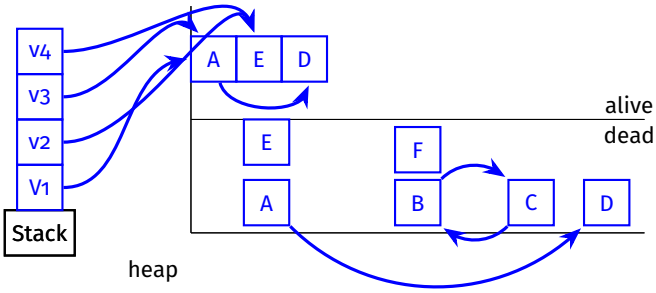


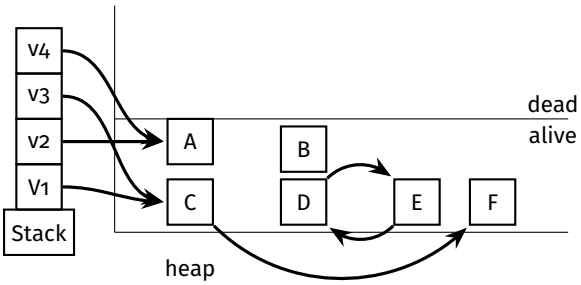
Suppose you have the following memory diagram. Draw the stack and heap after the stop and copy garbage collection technique is run. Note that nothing is popped off the stack for this question.

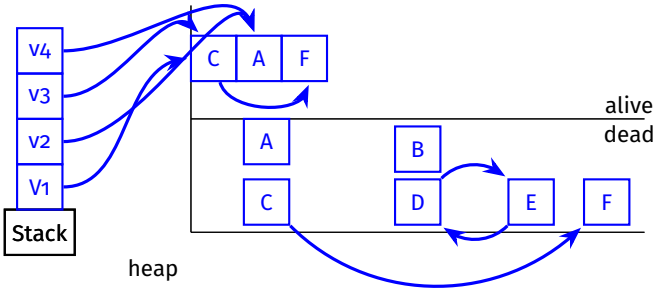


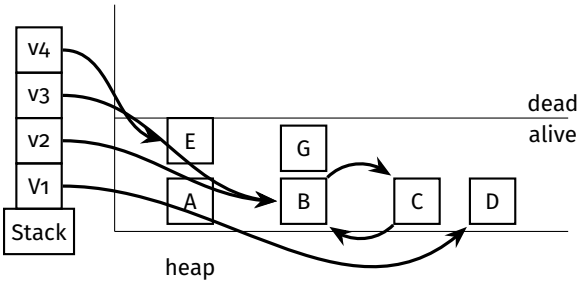


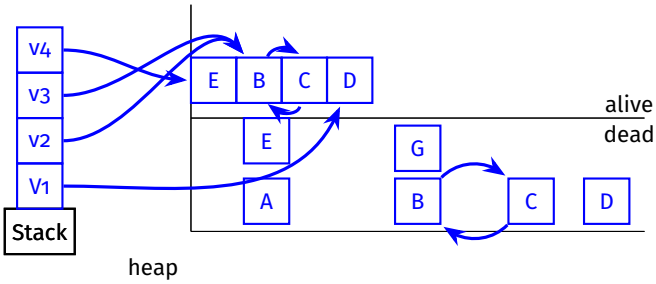












Problem 3: Rust Ownership and references

[Total 8 pts]

```
fn main(){
  {
    let mut a = String::from("hello");
    let mut b = String::from("World");
    // Mark 1
    {
      let c = f1(a);
      // Mark 2
      println!("{c}");
    }
    // Mark 3
    let d = f2(&mut b);
    println!("{b}");
    // Mark 4

    b = String::from("Bye");
    println!("{b}");
  }
}

fn f1(mut s: String) -> String{
  s.push_str(" Planet");
  // Mark 5
  s
}

fn f2(x: &mut String)-> usize{
    x.push_str(" domination");
    // Mark 6
    x.len()
}
```

If there is no owner (because the value has been dropped) put "None". Assume that we are asking about ownership **during** execution.

Who is the owner of the String with contents "hello" at Mark 1?

a

Who is the owner of the String with contents "hello Planet" at Mark 2?

c

Who is the owner of the String with contents "hello" at Mark 3?

None

Who is the owner of the String with contents "hello Planet" at Mark 5?

s

Who is the owner of the String with contents "World domination" at Mark 4?

b

Who is the owner of the String with contents "World domination" at Mark 6?

b

What is printed out when this program is run?

hello Planet

World Domination

Bye

```
fn main(){
    let mut a = String::from("best");
    let mut b = String::from("330");
    // Mark 1
    {
        let c = f1(&mut a);
        // Mark 2
        println!("{c}");
    }
    // Mark 3

    let d = f2(b);
    println!("{d}");
    // Mark 4

    a = String::from("it nothing beats!");
    println!("{a}");
}

fn f1(x: &mut String) -> &mut String {
    x.push_str(" class");
    // Mark 5
    x
}

fn f2(mut s : String) -> String {
    s.push_str(" is");
    // Mark 6
    s
}
```

If there is no owner (because the value has been dropped) put "None". Assume that we are asking about ownership **during** execution.

Who is the owner of the String with contents "best" at Mark 1?

a

Who is the owner of the String with contents "best class" at Mark 2?

a

Who is the owner of the String with contents "best class" at Mark 3?

a

Who is the owner of the String with contents "best class" at Mark 5?

a

Who is the owner of the String with contents "330 is" at Mark 4?

d

Who is the owner of the String with contents "330 is" at Mark 6?

s

What is printed out when this program is run?

best class

330 is

it nothing beats!

```
fn main(){
  {
    let mut a = String::from("hello");
    let mut b = String::from("World");
    // Mark 1
    {
      let c = f1(&mut a);
      // Mark 2
      println!("{c}");
    }
    // Mark 3
    let d = f2(b);
    println!("{d}");
    // Mark 4

    a = String::from("Bye");
    println!("{a}");
  }
}

fn f1(s: &mut String) -> &mut String{
  s.push_str(" Planet");
  // Mark 5
  s
}

fn f2(mut x: String)-> usize{
    // there is one single space before "domination"
    x.push_str(" domination");
    // Mark 6
    x.len()
}
```

If there is no owner (because the value has been dropped) put "None". Assume that we are asking about ownership **during** execution.

Who is the owner of the String with contents "World" at Mark 1?

b

Who is the owner of the String with contents "hello Planet" at Mark 2?

a

Who is the owner of the String with contents "hello Planet" at Mark 3?

a

Who is the owner of the String with contents "hello Planet" at Mark 5?

a

Who is the owner of the String with contents "World domination" at Mark 4?

None

Who is the owner of the String with contents "World domination" at Mark 6?

x

What is printed out when this program is run?

hello Planet

16

Bye

```
fn main(){
    let mut a = String::from("best");
    let mut b = String::from("330");
    // Mark 1

    {
        let d = f2(&mut b);
        // Mark 2
        println!("{d}");
    }
    // Mark 3

    let c = f1(a);
    // Mark 4
    println!("{c}");

    a = String::from("it nothing beats!");
    println!("{a}");
}

fn f1(mut s : String) -> String {
    s.push_str(" is");
    // Mark 5
    s
}

fn f2(x: &mut String) -> &mut String {
    x.push_str(" class");
    // Mark 6
    x
}
```

If there is no owner (because the value has been dropped) put "None". Assume that we are asking about ownership **during** execution.

Who is the owner of the String with contents "best" at Mark 1?

a

Who is the owner of the String with contents "330 class" at Mark 2?

b

Who is the owner of the String with contents "330 class" at Mark 3?

b

Who is the owner of the String with contents "best is" at Mark 5?

s

Who is the owner of the String with contents "best is" at Mark 4?

c

Who is the owner of the String with contents "330 class" at Mark 6?

b

What is printed out when this program is run?

330 class

best is

it nothing beats!