Student Name:

Student ID:

Midterm2 for CSE216: Programming Abstractions, State University of New York, Korea, 2024 Fall

- Maintaining academic integrity is critical.
- Any kind of electronic device is strictly prohibited during the exam, including iPads, Apple Watches, Samsung tablets, etc.
- You are allowed to use as many paper notes as you need.
- Ensure your handwriting is clear and legible. Any illegible answers may result in a score of zero.
- Be sure to submit the physical copy of your exam once you've finished.

Total Points = 108 (8 points are considered as extra credit)

Problem 1. OCaml Types (Points = 48. No partial points.)

Give the type of the following OCaml expressions:

1. `["hello"; "world"]`
2. `[[1]; [1]]`
3. `(1, "bar")`
4. `[(1, 2, "foo"); (3, 4, "bar")]`
5. `let f x y z = x + y + z in f 1 2`

Give the type of the following functions:

6. `let f x = x *. 3.14`
7. `let f (x, y) = x + y`
8. `let f x y = if y then x else x`
9. `let f x y z = if x then y else y`
10. `let rec f x = if (x = 0) then 1 else 1 + f (x - 1)`

11. `let alwaysfour x = 4`
12. `let add x y z = x + y + z`
13. `let addmult x y = (x + y, x * y)`
14. `let b (x, y) = x + y`
15. `let c (x, y) = x`
16. `let d x = match x with (a, b) -> a`

OCaml Functions (Points = 60. Partial points allowed.)

1. Write an OCaml function `lucas : int -> int` that calculates the nth number in the sequence of *Lucas numbers*. Lucas numbers begin with 2 and 1, and each subsequent number is the sum of the two preceding it. Assume the sequence starts from the 0-th number, and the input is a non-negative integer.

```
# lucas 0;;
- : int = 2
# lucas 1;;
- : int = 1
# lucas 10;;
- : int = 123
```

2. Implement an OCaml function `rev : 'a list -> 'a list` that computes the reverse of a list without using `List.rev`.

```
# rev [5; 6; 7];;
- : int list = [7; 6; 5]
```

3. Write a function `last : 'a list -> 'a option` that returns the last element of a list.

```
# last ["a"; "b"; "c"; "d"];;
- : string option = Some "d"
# last [];;
- : 'a option = None
```

Recall: In OCaml, functions that might not always return a value use the `option` type for safe handling. An `option` type can be:

- `Some value` – indicating a successful result with a `value`
- `None` – representing the absence of a value

This is useful when a function may not produce a meaningful value, such as when the list is empty. Using `option` allows the function to explicitly return `None` if no "last element" exists.

4. Write an OCaml function `compress : 'a list -> 'a list` that eliminates consecutive duplicates of list elements.

```
# compress ["a"; "a"; "a"; "a"; "b"; "c"; "c"; "a"; "a"; "d"; "e";
"e"; "e"; "e"];;
- : string list = ["a"; "b"; "c"; "a"; "d"; "e"]
```

5. Define the following OCaml functions:

   - `contains : 'a -> 'a list -> bool` that returns `true` if and only if the list contains the specified element. Do not use any pre-existing functions.

     ```
     # contains 4 [3; 4; 5];;
     - : bool = true
     ```

6. Define the following OCaml functions:

   - `evens : 'a list -> 'a list` which returns the 0th, 2nd, 4th, etc., elements of a list.

```
# evens [13; 5; 9; 0; 7; 8];;
- : int list = [13; 9; 7]
```