1.

2.

3. Hãy nêu điểm khác biệt chính của các cơ chế gọi chương trình con *đệ quy* (recursive), *biến cố* (exception), *trình cộng hành* (coroutine), *trình định thời* (scheduled subprogram) và *công tác* (task) so với cơ chế *gọi-trở về đơn giản* (simple call-return).

4. Cho một đoạn chương trình được viết trên ngôn ngữ tựa C như sau:

```
int A[3] = {1, 9, 45}; // index of A start from 0
int j = 0;
int n = 3;

int sumAndIncrease(int a, int i) {
    int s = 0;
    for ( ; i < n; i = i + 1) {
        s = s + a;
        A[j] = A[j] + 1;
    }
    return s;
}

void main() {
    int s = sumAndIncrease(A[j], j);
    cout << s << A[0] << A[1] << A[2]; // 1
}
```

a) s = 4
A[0] = 4
A[1] = 9
A[2] = 45
b)
s = 5
A[0] = 2
A[1] = 10
A[2] = 46
c)
s = 55
A[0] = 2
A[1] = 10
A[2] = 46

Hãy cho biết và giải thích kết quả in ra của chương trình trong các trường hợp sau:

(a) Nếu a và i được truyền bằng trị-kết quả.

(b) Nếu a và i được truyền bằng tham khảo.

(c) Nếu a và i được truyền bằng tên.

5. Cho đoạn mã sau được viết trên ngôn ngữ **quy tắc tầm vực tĩnh** (static-scope rule):

```
procedure main() {
    var a, b, c: integer; // 1
    procedure sub1(a: integer) { // 2
        procedure sub3();
        procedure sub2() {
            var a, c: real; // 3
            sub3();
        }
        procedure sub3() {
            a   // use a
        }
        sub2();
    }
    sub1(3);
}
```

a) main: a,b,c,sub1,main
sub1: a//2,b,c sub1,sub3,sub2,main
sub2: b, a//2,c //3 ,sub3, sub2,main
sub3: a//2, bc//1, main,

b) a//1: main
b//1: main, sub1, sub2, sub3,
a//2: sub1, sub3, sub2
sub2: sub2. sub1, sub3
a//3: sub2
sub3: sub1,sub2, sub3

(a) Cho biết môi trường tham khảo tĩnh (static referencing enviroment) cho các thủ tục main, sub1, sub2, sub3.

(b) Cho biết tầm vực tĩnh của các khai báo a//1, b//1, a//2, sub2, a//3, sub3.

6. Cho khai báo sau được viết trên ngôn ngữ lập trình Ada:

```
type Shape is (Circle, Triangle, Rectangle);
type Colors is (Red, Green, Blue);
type Figure (Form: Shape) is record
    Filled: Boolean;
```

```
    Color: Colors;
    case Form is
        when Circle => Diameter: Float;
        when Triangle =>
            Leftside, Rightside: Integer;
            Angle: Float;
        when Rectangle => Side1, Side2: Integer;
    end case;
end record;
```

(a) Vẽ mô hình mô tả một đối tượng (các thành phần, kích thước các thành phần) có kiểu Figure và qua đó tính tổng kích thước của một đối tượng kiểu Figure. Giả sử kích thước của kiểu Integer, Float, Boolean và Enumeration lần lượt là 4, 4, 1 và 4. Cần chú ý về vấn đề padding.

(b) Trên những ngôn ngữ lập trình không có kiểu union như Java, Scala, C# thì làm thế nào để thực hiện một đối tượng kiểu union (như Figure)? Hãy thực hiện kiểu Figure trên một trong các ngôn ngữ lập trình Java, Scala, C#.

7. Cho đoạn mã sau:

```
int p;
int* foo(int x) {
    static int q;
    int* s = new int;
    switch(x) {
        case 1: return &p;
        case 2; return &q;
        case 3: return &x;
        case 4: return s;
        default: return foo(x-1);
    }
}
```

(a) Hãy xác định và giải thích các lỗi liên quan đến con trỏ (*tham chiếu treo* - dangling reference, *rác* - garbage) có thể xảy ra khi thực thi đoạn mã trên.

(b) Khi hàm foo được gọi đệ quy thì các bản hoạt động của foo dùng chung những đối tượng dữ liệu nào (p, q, s, x, đối tượng được tạo ra bởi new int)? Giải thích.

8. Chuyển sang dạng tiền tố (Cambridge Polish Prefix):

$$a * b * c + d + e - f$$

Yêu cầu: cùng thứ tự xuất hiện toán hạng. Số dấu '(' và ')' ít nhất. Cùng thứ tự tính toán.

9. `def lookup[T](x: String, lst: List[T], f: T =>String): Option[T]`
Với `x` là chuỗi cần tìm kiếm, `lst` là danh sách để thực hiện tìm kiếm, `f` là hàm để trích xuất thành phần kiểu chuỗi trong mỗi phần tử của `lst` để so sánh với `x` khi tìm kiếm.
Hãy hiện thực (viết thân hàm) `lookup` trên bằng ngôn ngữ Scala để thực hiện tìm kiếm một chuỗi trên danh sách `lst` với hàm `f`.

10. Hãy hiện thực (viết các chương trình thể hiện các ràng buộc kiểu) suy diễn kiểu để suy ra kiểu của hàm sau:

```
H(x, f, h) {
  s := 0
  for i := f(x) to h(x) do s := s + i
  return s
}
```

Các ràng buộc trên các phát biểu gán, for, phép toán + tương tự như trên BKOOL