Câu 1: 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)

Trả lời: Cơ chế gọi trở về đơn giản có các đặc điểm sau:

+ Không có đệ quy (Khác với cơ chế gọi chương trình con đệ quy: có thể đệ quy)
+ Lệnh gọi tường mình (Khác với exception : không có lệnh gọi tường minh)
+ 1 điểm vào (Khác với coroutine : có nhiều điểm vào)
+ Chuyển điều khiển ngay lập tức (Khác với scheduled subprogram : không chuyển được ngay)
+ Thực thi đơn (Khác với task: thực thi đa)

câu 2:

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
}
```

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.

a)
Kết quả trả về : 3 4 9 45
Giải thích :
Vì a và i được truyền bằng trị- kết quả nên ban đầu:
a và i sẽ nhận giá trị lần lượt là a = 1 và i =0;
Khi thực hiện các lệnh của hàm sumAndIncrease:
giá trị của các thành phần như sau:
s = 3;
A[3] ={4,9,45}

Khi kết thúc chương trình con, giá trị của a và i sẽ được gán trở lại cho A[j] và j (Tức là A[0] và j). Khi đó : A[0] = 1

Vậy kết qua trả về là : 3 1 10 46

b)

Kết quả trả về : 6 4 9 45

Giải thích:
Nếu a và i được truyền tham khảo, thì khi đó a và i sẽ trỏ đến vùng nhớ của A[0] và j.
khi đó, mọi thay đổi giá trị của a và i sẽ trực tiếp thay đổi giá trị cùa vùng nhớ của A[0] và j và ngược lại.
Ở vòng lặp thứ nhất (i=0), ta có:
s = s + a (a =1, s = 1)
A[j] = A[j] + 1 ; (A[0] =2 => a =2)
Ở vòng lặp thứ hai (i=1), ta có:
s = s + a; (a=2, s= 3)
A[j] = A[j] +1; (A[0] = 3 => a = 3)
Ở vòng lặp thứ ba (i=2), ta có:

s = s+ a; (a=3, s=6)
A[j] = A[j] +1; (A[0] = 4 => a =4)

c)
Kết quả trả về:55 2 10 46

nếu a và i được truyền bằng tên, khi đó ta thay thế a thành A[j], và i thành j.
Khi đó hàm SumAndIncrease sẽ trở thành:

```
int s =0;
for ( ; j < n ; j = j+1){
        s = s + A[j];
        A[j] = A[j] +1;
}
```

với vòng lặp thứ nhất (j =0), ta có :
s = s + A[j] (s = 1)
A[j] = A[j] +1 (A[0] = 2)

Với vòng lặp thứ hai (j=1), ta có:
s = s + A[j] (s= 1 + 9 = 10)
A[j] = A[j] +1 (A[1]= 10)
Với vòng lặp thứ ba (j=2), ta có:
s = s +A[j] (s = 55)
A[j] = A[j] +1 (A[2] = 46)

Câu 3:

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) 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.

a)

Function	Referencing enviroment
main	a//1, b//1, c//1, sub1
sub1	a//2, b//1, c//1, sub1, sub2, sub3
sub2	a//3, c//3, b//1, sub1, sub2, sub3
sub3	a//2, b//1, c//1, sub1, sub2, sub3

b)

a//1	maib
b//1	main, sub1,sub2,sub3
a//2	sub1,sub3
a//3	sub2
sub2	sub1, sub2, sub3
sub3	sub1, sub2, sub3

Câu 4:

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;
```

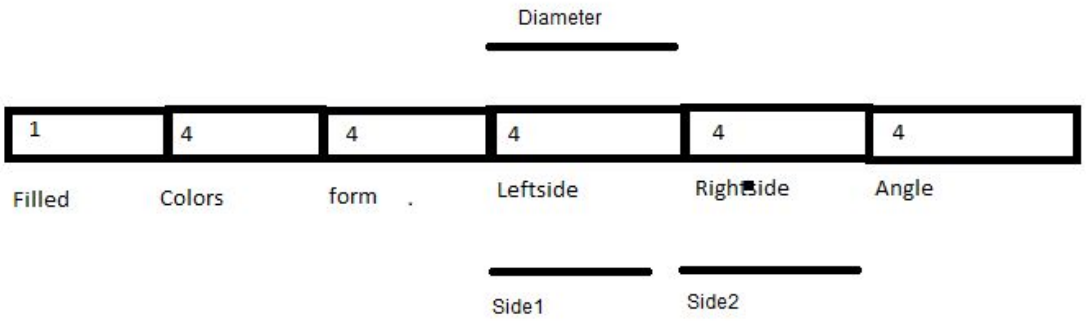


```
   Color: Colors;
   case Form is
      when Circle => Diameter: Float;
      when Triangle =>
         Leftside, Rightside: Integer;
         Angle: Float;
      when Rectangle => Side1, Side2: Integer;
   end case;
end record;
```

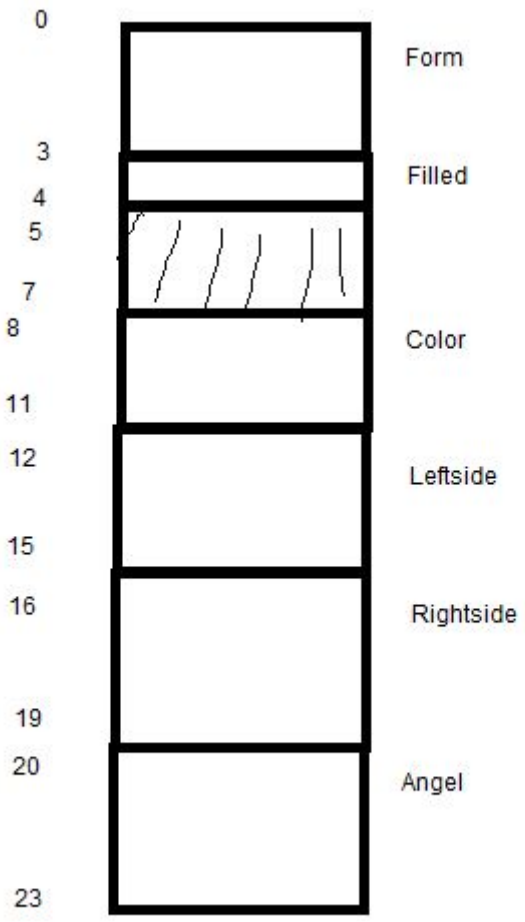
(a) Vẽ mô hình mô tả một đối tượng (các thành phần, kích thước các thành phần) có kiểu Figure và qua đó tính tổng kích thước của một đối tượng kiểu Figure. Giả sử kích thước của kiểu Integer, Float, Boolean và Enumeration lần lượt là 4, 4, 1 và 4. Cần chú ý về vấn đề padding.

(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#.

a)
Với trường hợp bỏ qua padding :



Với trường hợp xét padding:



b) Sử dụng Kế thừa thay cho dùng Union

```
Class Figure {
        bool Filled;
        Colors color;
}
Class Circle extends Figure{
        float Diameter;
}
Class Triangle extends Figure{
        int Leftside;
        int Rightside;
        float angle;
}
Class Rectangle extends Figure{
        int side1;
        int side2;
}
```

Câu 5

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.

Cambridge Polish Prefix : (-(+(*abc)de)f)

Câu 6:

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`.

Làm bằng python:

```
def lookup(self,x,lst,f):
        for y in lst:
            if x == f(y):
                return y
        return None
```

Câu 7:

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

H (T1 xT2x T3) -> T4:

+ xét f(x):
  T1 -> T2 ;
  Vì i là kiểu int nên T2 cũng là kiểu int
+ xét h(x):
  T1 -> T3
  vì trong vòng for nên h(x) phải là kiểu int => T3: int
+ Vì i là kiểu int, nên s cũng là kiểu int => T4 : int
+

vậy :
(T1 x (T1 -> int)x (T1 -> int)) -> int