Ejercicio 1: Pseudo código

```
function calcularTerminanAntes_ai (S , F, i)
    S_i = [ ]
    F_i = [ ]
    for l=0 to s.length-1 do
        if s[i] ≥ f[l] then
            S_i.append [S[l])
            F_i.append [F[l])
        end if
    end for
    return S_i, F_i
end function

function calcularInicianDespués_ai (S, F, i)
    S_i = [ ]
    F_i = [ ]
    for l=0 to S.length-1 do
        if s[l] ≥ f [i] then
            S_i.append [S[l])
            F_i.append [F[l])
        end if
    end for
    return S_i, F_i
end function
```

```
function SelecciónActividades_valoropts[s,f]
    if s.length==0 then
        return 0
    end if
    n = s.length
    q = -1
    for k=0 to n-1 do
        ST_k, FT_k = calcularTerminanAntes_ai(s,f,k)
        SD_k, FD_k = calcularIncianDespues_ai[s,f,k]
        p = 1 + SeleccionActividades_valoropt(ST_k, FT_k) +
            SeleccionActividades_valorOpt(SD_k, FD_k)
        if p>q then
            q=p
        end if
    end for
    return q
end function
```

Ejercicio 2 : Correctez

Algoritmo Termina

El programa tiene un for que va de 0 a n-1 donde n es la longitud del arreglo de tiempos de actividades. Al salir del for del algoritmo regresa q, por lo que concluimos que el algoritmo termina.

Invariante de ciclo

Al final de la k-ésima iteración, la variable q contiene el máximo número de actividades con la actividad k

Prueba la correctez

Inicialización

Al inicio de ciclo, el conjunto de actividades compatibles con k se encuentra vacio, entonces podemos concluir que la invariante se cumple

Mantenimiento

Para este caso, se debe probar por inducción que la invariante se cumple. Suponemos que el invariante es cierto en la iteración k y se probará para la iteración $k+1$.

En la k-ésima iteración los conjuntos de actividades ST_k, FT_k, SD_k y FD_k corresponden a las actividades que terminan antes de que inicie k, o inician después de que k termina. Este es el conjunto de actividades compatibles con k. Denotemos el conjunto $S_k = \{ST\_k, FT\_k, SD\_k, FD\_k\}$

En la línea 49 calculamos de forma recursiva el número de actividades compatibles con k. Esto tambien sucederá con el conjunto $S_{k+1} = \{ST\_k+1, FT\_k+1, SD\_k+1, FD\_k+1\}$ si el número de actividades en S_k+1 es mayor al número de actividades en S_k, este número será el

máximo(nuevo) de actividades compatibles entre si. De esta forma, se cumple el invariante.

Terminación

Por la condición de mantenimiento, en la ultima iteración también se validarian los valores de p y q verificando si p>q, que en caso que se cumpla, se actualizaria el número máximo, compliendo el invariant