Problema 1.

a. Recordar que, $8 = 2^3$

Basta con agrupar de a 3 bits y tomar su representación en base 10 para cada grupo:
$10010000111010000010001_2$

2207202_{18}

b. 123400_6

Recordar que: $1234_6 = 1*6^3 + 2*6^2 + 3*6^1 + 4*6^0$
$100_6 = 1*6^2 + 0*6^1 + 0*6^0 = 1*6^2$

Entonces, $1234_6 * 100_6 = (1*6^3 + 2*6^2 + 3*6^1 + 4*6^0)*(6^2) = (1*6^5 + 2*6^4 + 3*6^3 + 4*6^2) = 123400_6$

c. En complemento a 1, donde 1110 representa a -1 y 0010 al 2, entonces el resultado de sumarlos es 0001 que corresponde a 1.

d. 0000 (0)
1111 (-1)
1110 (-2)
1101 (-3)
1100 (-4)
1011 (-5)
1010 (-6)
1001 (-7)
1000 (-8)
el (-9) no se puede representar correctamente en con este sistema de numeración y dicha cantidad de bits.

e. Representar el número 2.5 utilizando punto flotante con la siguiente distribución de bits:
1s 4e 7m
Signo: Como 2.5 > 0, entonces s = 0
Mantisa y exponente:

$2.5 = 10.1_2 * 2^0 = 1.01_2 * 2^1$ (Corremos un lugar la coma para luego poder omitir el bit más significativo)
Tenemos entonces que, la mantisa m = **01**00000
Entonces tenemos que el exponente será 1 más el exceso M, e = 1 + M = 1000

El resultado:
010000100000

Problema 2.

Notar que b queda determinado por b = funcAux(b), donde el b pasado como parámetro a la función vale 8, entonces b = funcAux(8) y luego a = a – 1 determina el valor de a (la variable a de la función no afecta a la variable a del script), quedando a = 3 – 1 = 2.

Veamos la llamada de miscript.m b = funcAux(8) = 15

↘

8 + funcAux(6) = 8 + 7 = 15

↘

6 + funcAux(4) = 6 + 1 = 7

↘

1

Entonces, luego de ejectuar miscript.m las variables a y b valen a = 2 y b = 15.

Problema 3.

a) El resultado de invocar la función va a producir un error, ya que la función devuelve 2 parámetros de salida y en las invocaciones solamente se espera un resultado.

b)
```
function [pares, impares] = separar(v)
    largo = length(v);
    if (largo == 0)
        pares = [];
        impares = [];
    else
        [pares, impares] = separar(v(2:largo));
        if (mod(v(1),2) == 0)
            pares = [v(1) pares];
        else
            impares = [v(1) impares];
        end
    end
end
```

Problema 4.

```
function res = multiplosDe(v1,v2)
        res = 1;
        largo = length(v1);
        iter = 1;
        while (res == 1 && iter <= largo)
                if mod(v1(i),v2(i)) != 0
                        res = 0;
                end
                iter = iter + 1;
        end
end


function res = multiplosDe(v1,v2)
        res = 1;
        largo = length(v1);
        iter = 1;
        while (res == 1 && iter <= largo)
                res = (mod(v1(i),v2(i)) == 0);
                iter = iter + 1;
        end
end
```

Problema 5.

```
function [indice,valor] = mejorPivote(v)
	largo = length(v);
	indice = 1;
	valor = abs(v(1));
	for iter = 2:largo
		if abs(v(iter)) > valor
			valor = abs(v(iter));
			indice = iter;
		end
	end
end
```

Problema 6

```
function ker = nucleo(Ad,Af,Ac)
	ker = 0;
	cant_elem = length(Ad); %Puede ser cualquiera de los 3
	for iter = 1:cant_elem
		if (Af(iter) == Ac(iter)) % Corresponde a una entrada de la diagonal
			ker = ker + Ad(iter);
		end
	end
end
```

Problema 7.

```
function res = multiplosDe(v1,v2)
        largo = length(v1);
        if largo == 0
                res = 1;
        elseif mod(v1(1),v2(1)) == 0
                res = multiplosDe(v1(2:largo),v2(2:largo));
        else
                res = 0;
        end
end
```

Problema 8.

```
function ker = nucleo(Ad,Af,Ac)
        cant_elem = length(Ad);
        if cant_elem == 0
                ker = 0;
        else
                ker = nucleo(Ad(2:cant_elem),Af(2:cant_elem),Ac(2:cant_elem);
                if Af(1) == Ac(1)
                        ker = ker + Ad(1);
                end
        end
end
```

Problema 9.

```
function [indice,mejorValorAbs] = mejorPivote(v) % Resolución desde el primer elemento del vector
        largo = length(v);
        if largo == 1 %Caso base
                indice = 1;
                mejorValorAbs = abs(v(1));
        else
                [indice,mejorValorAbs] = mejorPivote(v(2:largo));
                indice = indice + 1;
                If abs(v(1)) > mejorValorAbs
                        mejorValorAbs = abs(v1);
                        indice = 1;
                end
        end
end
function [indice,mejorValorAbs] = mejorPivote(v) % Resolución desde el último elemento del vector
        largo = length(v);
        if largo == 1 %Caso base
                indice = 1;
                mejorValorAbs = abs(v(1));
        else
                [indice,mejorValorAbs] = mejorPivote(v(1:largo-1));
                If abs(v(largo)) > mejorValorAbs
                        mejorValorAbs = abs(v(largo));
                        indice = largo;
                end
        end
end
```