Heuristic Algorithms (4 points)

Name: Surname:

Note1: Write your answers exclusively in the space reserved.
Note2: Examples of the expected format of your answers are given just for illustrative purposes.

1.- GRASP Constructive (1.5 points)

We want to solve the Set Covering problem using GRASP and are given a universe of elements M and a family of subsets P={pj}j∈N, N={1, 2, … , n}.

For the constructive phase, we are considering the following greedy function:

$q(p_j|R)=|p_j \cap (M \backslash R)|$

being:

S the solution sub-family
R the set of covered elements

We are given the following instance of the problem:

M/P	p1	p2	p3	p4	p5	p6	p7	p8
1						X		
2			X	X			X	
3		X		X	X		X	X
4	X			X	X	X		X
5					X	X		

Imagine that we already have the following partial solution:

S:{p4}
R={2, 3, 4}

a) Assuming α=0.5. Compute the elements in the RCL.

Your answer (e.g., RCL = {p1, p2, p3, p6}:

RCL =

b) What would be the solution S and its cost (value of the objective function f()) after the constructive phase, assuming a purely greedy policy?

Your answer (e.g., S = {p1, p2, p3, p6}, f(S) = 8):

S = f(S) =

2.- Local Search (1 point)

We are implementing a Local Search phase for the Set covering problem and we are defining two neighborhoods:

- $N_0(S)$ = remove as many pj as possible from S.

- $N_1(S)$ = exchange one pj in S with another not in S.

M/P	p1	p2	p3	p4	p5	p6	p7	p8
1						X		
2			X	X			X	
3		X		X	X		X	X
4	X			X	X	X		X
5					X	X		

a) For the solution S = {p2, p3, p6, p8}, list the neighboring solutions S' in N_0.

Your answer (e.g., $N_0(S)$ = {{p2, p3, p6, p7}, {p2, p3, p6, p8}}):

$N_0(S)$ =

b) For the solution S = {p4, p6}, list the neighboring solutions S' in N_1 with f(S')<f(S).

Your answer (e.g., $N_1(S)$ = {{p2, p3, p6, p7}, {p2, p3, p6, p8}}):

$N_1(S)$ =

3.- BRKGA (1.5 points)

To solve large instances of the Travelling Salesman Problem (TSP), we have decided to use the BRKGA metaheuristic. For the decoder, we are adapting the nearest neighbor (NN) algorithm defined in the following pseudocode, where we are given a graph G(*V*,*E*):

```
Tour ← {v}, where v is an arbitrary node (starting point) from V
while Tour ≠ V do
        C ← set of feasible links w.r.t. Tour ⊆ E
        e_best ← (u ∈Tour, v ∉Tour) ← argmin{d(Tour, e) | e in C}
        Tour ← Tour U {v}
return Tour
```

by implementing the following modifications:

1) The decoder receives as input a chromosome with random keys, one for each edge *e*∈*E*.
2) Each edge is weighted with the corresponding gene in the chromosome and the resulting weights are used as distances in the modified NN algorithm.

Given the graph G(*V*,*E*) represented by the distance matrix:

	1	**2**	**3**	**4**	**5**
1	-	10	12	7	9
2		-	11	17	4
3			-	5	14
4				-	9
5					-

What would be the solution when the decoder receives as input the following chromosome:

Edge	**1-2**	**1-3**	**1-4**	**1-5**	**2-3**	**2-4**	**2-5**	**3-4**	**3-5**	**4-5**
Chromosome	0.9	0.2	0.2	0.3	0.4	0.1	0.1	0.9	0.1	0.9

Your answer (e.g., Tour = 1-2-3-2-1-6):

Tour =