Question 1 :

(a) p est strictement croissant donc p atteint n.

(b) n = 5 :

r | i | p

1 1 1

4 3 2

9 5 3

16 7 4

(c) i = (2\*p)-1

Posons P(i) = (2\*p)-1 :

Cas de base : i = 1 cet à dire P(1) = (2\*1)-1 = 2-1 = 1 donc

P(i) est vérifier pour i = 1.

Hérédité : Soit i € {1....n} un entier pour lequel P(i) est vérifié

montrons que P(i+1) est vraie.

P(i+1) = (2\*(p+1))-1

P(i) = (2\*p)-1

P(i)+1 = (2\*p)-1+1

P(i)+1 = (2\*p)

//Je rajoute +1 et je passe le premier +1 de l'autre côté.

P(i+1) = (2\*p+1)-1

(d) r = p²

Poson P(r) = p²

Cas de base : r = 1 cet à dire P(1) = 1² = 1 donc P(r) est vérifier

pour r = 1.

Hérédité : Soit r € {1....n} un entier pour lequel P(r) est vérifié

montrons que P(r+1) est vraie.

P(r+1)=(p+1)²

Soit P(r) = p²

P(r+1) = (p+1)² //C'est une identité.

P(r+1) = p² + 2\*(p)+1²

P(r+1) = p² + 2p + 1

(e) L'algorithme g calcul le carré de n-1, pour le prouver il suffit de voir

que p s'arrête 1 itération avant n cet à dire qu'a la fin p = n-1

on a prouver précédemment que r = p² donc à la dernière itération p = n-1 et

p² = r donc r = (n-1)².

Question 2 :

(a) Algorithme : quotient(d a : entier, d b : entier, r q: entier)

Données : a € N, b € N\*

Résultat : le quotient de a par b.

i <- a; j<- b; q<-0;

Tant que i >= j faire

| i <- i-j;

| q <- q+1;

fin Tant que

renvoyer q;

i | j | q

17 3 0

14 3 1

11 3 2

8 3 3

5 3 4

2 3 5

(b)

1) i-j est strictement décroissant donc i devient plus petit que j

et la condition de boucle est atteinte.

2) Invariant : i = (i+1)+j

3) Si (i+1)+j est bien égal à i cet à dire qu'à l'étape i+1 i a bien

été décrémenté de la valeur j donc que le tour de boucle c'est bien passé

et que q a été incrémenté.

Question 3 :

(a)

Debut<-0; Fin<-taille(T)-1; Milieu<-(Debut+Fin)/2; Max<-T[Debut]

Tant que (Deb <= Fin)

Milieu<-(Debut+Fin)/2;

Si Max < T[Milieu] alors

Debut<-Milieu;

Max<-T[Milieu];

Sinon

Fin<-Milieu-1;

fin si

fin Sinon

fin tant que

renvoyer Max;

(b)

Invariant : Debut < Milieu <= Fin

Compléxité : 0(log2(n))