PARTIE 1 (LOAD FACTOR = 0.50)

	TEMPS	CONFLICTS	NB AMAS	TailleMinAmas	TailleMaxAmas	TailleMoyenAmas
Size15: LinearHashTable :	54300	7	11	1	2	1.3636363636363635
Size15: QuadraticHashTable :	41000	8	11	1	2	1.3636363636363635
Size15: DoubleHashTable :	56800	7	11	1	2	1.3636363636363635
Size60: LinearHashTable :	102800	51	34	1	7	1.7647058823529411
Size60: QuadraticHashTable :	127900	44	35	1	7	1.7142857142857142
Size60: DoubleHashTable :	335100	44	37	1	6	1.6216216216216217
Size150:LinearHashTable :	291000	151	72	1	12	2.0833333333333335
Size150:QuadraticHashTable :	158500	125	76	1	7	1.9736842105263157
Size150:DoubleHashTable :	558300	130	75	1	17	2.0

On remarque que plus le nombre d'éléments à insérer augmente plus il y a de collisions. On peut aussi voir que la LinearHashTable a un nombre de collisions qui augmente plus rapidement que les autres tables à force d'augmenter le nombre d'insertions.

De plus, l'insertion des N éléments devrait être effectué avec une complexité temporelle linéaire et cela est confirmé par nos résultats puisque plus on insert d'éléments plus le temps d'exécution est grand.

PARTIE 2

LOAD FACTOR = 0.25

	TEMPS	CONFLICTS	NB AMAS	TailleMinAmas	TailleMaxAmas	TailleMoyenAmas
Size15: LinearHashTable :	56300	3	14	1	2	1.0714285714285714
Size15: QuadraticHashTable :	31500	3	14	1	2	1.0714285714285714
Size15: DoubleHashTable :	34500	3	14	1	2	1.0714285714285714
Size60: LinearHashTable :	119900	17	47	1	3	1.2765957446808511
Size60: QuadraticHashTable :	87000	17	47	1	3	1.2765957446808511
Size60: DoubleHashTable :	385300	17	49	1	3	1.2244897959183674
Size150:LinearHashTable :	272500	50	109	1	4	1.3761467889908257
Size150:QuadraticHashTable :	188600	47	112	1	3	1.3392857142857142
Size150:DoubleHashTable :	874600	51	115	1	4	1.3043478260869565

LOAD FACTOR = 0.75

	TEMPS	CONFLICTS	NB AMAS	TailleMinAmas	TailleMaxAmas	TailleMoyenAmas
Size15: LinearHashTable :	64600	24	5	1	9	3.0
Size15: QuadraticHashTable :	45300	17	5	1	9	3.0
Size15: DoubleHashTable :	39800	15	6	1	8	2.5
Size60: LinearHashTable :	123100	107	19	1	20	3.1578947368421053
Size60: QuadraticHashTable :	75200	89	19	1	13	3.1578947368421053
Size60: DoubleHashTable :	266900	89	20	1	16	3.0
Size150:LinearHashTable :	470400	402	72	1	12	2.0833333333333335
Size150:QuadraticHashTable :	191400	280	78	1	10	1.9230769230769231
Size150:DoubleHashTable :	694100	333	77	1	13	1.948051948051948

Avec les 3 tableaux de load factor différent, on remarque que plus le load factor augmente, plus le nombre de collisions augmente.Donc, plus le load factor est petit, plus la fonction rehash est appelé souvent pour un certain nombre d'insertion et cela augmente le nombre d'espace disponible pour les nouvelles valeurs, ce qui diminue le nombre de collisions. Cependant, l'espace mémoire alloué sera plus grand lorsque le load factor est petit.

SANS REHASH LORQU'ON APPEL LES HASH TABLES AVEC UN SIZE = À LA TAILLE DE L'ARRAY INSÉRÉ :

	TEMPS	CONFLICTS	NB AMAS	TailleMinAmas	TailleMaxAmas	TailleMoyenAmas
Size15: LinearHashTable :	47600	14	2	4	11	7.5
Size15: QuadraticHashTable :	17500	10	2	4	11	7.5
Size15: DoubleHashTable :	23000	9	2	5	10	7.5
Size60: LinearHashTable :	83900	110	2	13	47	30.0
Size60: QuadraticHashTable :	60800	155	2	13	47	30.0
Size60: DoubleHashTable :	137700	155	2	5	55	30.0
Size150:LinearHashTable :	281700	924	2	12	138	75.0
Size150:QuadraticHashTable :	88600	471	2	7	143	75.0
Size150:DoubleHashTable :	618100	730	2	11	139	75.0

La complexité asymptotique du rehash est O(n) puisqu'on recopie l'ensemble les n éléments de l'ancienne array dans une nouvelle plus grande.

Selon nos résultats, le temps d'exécution sans rehash est plus rapide surement à cause qu'on ne fait plus aucun appel à rehash ce qui fait en sorte qu'il y a moins d'opérations de copie d'array de complexité O(n).

Le nombre de conflits est clairement supérieur sans le rehash puisque la table a une taille fixe et donc au moment que le load factor dépasse 50% aucun rehash est exécuté et donc le nombre de collisions augmente de façon exponentielle.