Ex 1

(drop(n)(take(n+1)x)) fait presque la meme chose comme [(x!!n)] et ca retourne False au lieu de "Exception: Prelude.(!!): index too large" si l'indice depasse.

Ex 2

Dans ce code, on verifie la taille des deux strings pour pas deborder. Sinon ca retourne -1. Pendant la comparaison, on compare chaque lettre du string, indice par indice, par recursivite. Si la lettre de chaque string x et y sont egaux, alors on retourne le numero de l'indice n.

Ex 3

On verifie d’abord si le string est vide. S’il est, on retourne string vide. Par recursivite, on prend la queue du chaine et on ajoute la tete a la fin. La tete de la queue va etre devant de la queue jusqu’a qu’on atteint une chaine vide. Au final, la chaine au complete est inversee.

Ex 4

On verifie si la taille du string est vide au debut. Si ca equivaut a 1 chaine de caractere, on retourne la caractere. Si le premier et le dernier caracteres ne sont pas egal, on retourne un string vide. Si la tete et l’inverse de la tete sont egaux, on ajoute le caractere de la tete et la queue recursivement et on continue ajouter la queue jusqu’a ce que la tete et son inverse ne sont plus egaux.

Ex 5

On prend en parametre la valeur x. On effectue x modulo 2. Si ca equivaut a 1, alors le nombre est impaire. Si la liste est vide, on retourne un liste vide. Si le nombre est pair, alors on n’affiche pas dans la liste.

Ex 6

Si la liste est vide, on retourne la liste vide. On affiche la liste des nombres impaires d’abord et ensuite on concartenere la liste inversee des nombres pairs.

Ex 7

On a la chaine xx. Si elle est vide, alors on retourne le resultat vide. Dans la fonction crypter, on prend la tete de la chaine, qui devient une chaine de caractere, et on utilise fromEnum pour additionner son numero ASCII au nombre n. On le reconvertir celui-ci en Char. Le recursivite va prendre la queue de la chaine xx et va effectuer la meme procedure. La fonction decrypter suit la meme procedure que la fonction crypter, mais au lieu d’additionner, on soustraire par le nombre n. Dans la fonction crypter\_decrypter, si le premier parametre c’est E, on effectue un cryptage. Si c’est R, on effectue un decryptage. Si ce n’est ni E ou R, on obtient un message erreur “Operation non autorisee”.

Ex 8

Pour la distance Euclidienne, on cree un nouveau fonction e\_dist\_somme qui calcule la somme des points (v2-v1)^2 + ... (vs2-vs1)^2 par recursivite. Ensuite, pour la fonction euclidean\_dist, on fait la racine carree de e\_dist\_somme. Pour les 2 fonctions, il faut verifier si les 2 listes sont vides. S’ils sont vide, on retourne 0.0. La fonction dist\_lst calcule la distance Euclidienne d’un vecteur pour chaque vecteurs dans lst (type [Knn\_data]) a l’aide de la recursivite. La plus petite valeur represente la distance Euclidienne la plus courte. Pour la fonction knn, on verifie d’abord si la fin d’une liste est vide. Alors on retourne un chaine vide. On prend (dist\_lst vect [k]) qui est une distance euclidienne calculee et on compare a la valeur minimum dans (dist\_lst vect (k:ks)). Si les deux valeurs ne sont pas equivalent, on effectue un recursivite jusqu’a temps les deux valeurs sont equivalents. Lorsque c’est equivalent, on affiche son String approprie “classe1”, “classe2” ou “classe3”.