Rendu Semaine 12 et 13

David Wiedemann

7 décembre 2020

2.4.1

Un type possible est

```
struct Paquet{
bool donnee;
int seq;
int data
};
```

2.4.2

```
bool bon_ordre(Paquet A,Paquet B) {
if( A.seq< B.seq) {
return true;
}
return false;
}
```

2.4.3

Je suppose la liste L non vide, il suffirait de return un entier quelconque si c'etait le cas.

```
int recevoir_TCP(Paquet P, int i, vector<Paquet> L){
if( P.donnee==false) {
return i;
}
Paquet acquit;
acquit.data= P.data;
acquit.seq = P.seq;
acquit.donnee= false;
```

```
send( acquit);

if( P.seq != i+1) {
insert( P) ;
return i;
}
affiche_paquet( P);
if(L[0].seq+1==P.seq) {
int i( 0);
while(L[i].seq-1== L[i+1].seq && i<L.size( )-1 ) {
affiche(L[i]);
++i;
}
}
return i;
}
```

2.3.2

Le message, une fois encrypte, sera de la forme

$$238^{79} \mod 889$$

Et, pour decrypter l'entier 532, on trouvera

$$532^{319} \mod 889$$

2.3.3

Etant donne qu'on ne se soucie pas de complexite algorithmique, une fonction avec une complexite en ordre lineaire semble suffisante...

```
int exp_et_modulo( int x, int y, int z) {
int n=1;
for( int i( 0);i<y;++i) {
        n*=x;

}
return n%z;
}
```