IPST-Cnam 21/03/2019

Licence Informatique Générale - Examen *RSX102* – 1ère session

B.RAIFF

Durée : 2H30 – Tous documents autorisés, mais pas les outils de communication (ordinateurs, téléphones)
Ne cherchez pas forcément à tout traiter, faites votre choix dans les questions.
Barème indicatif (le total des points obtenus (sur un maximum de 30) donnera directement la note sur 20)
Soyez concis et rigoureux dans le vocabulaire utilisé.
Lire attentivement la question avant de tenter d'y répondre, pour éviter les hors sujets !
Toutes les réponses résultant d'un calcul doivent être justifiées par le calcul correspondant.
S'il vous manque une hypothèse pour répondre à une question, faites-là en la justifiant.

Exercice 1 : Sockets IP *(5 points)*

1. Une machine Linux d'@IP (A), **administrable en SSH**, fait office de **serveur Web, SMTP et MySQL**
2. Une machine (B) consulte au moment de l'étude le site Web de **A** (avec un navigateur)
3. **A** consulte au moment de l'étude une page Web "complexe" sur un serveur web **C** et pour cela a ouvert simultanément **2 (deux)** connexions vers (C)
4. **A** a lancé un client FTP et procède au moment de l'étude à un (gros) téléchargement (**download**), en mode **actif**, à partir d'un serveur FTP situé en (D) (lisez bien cette phrase)
5. **A** a lancé un deuxième client FTP et procède au moment de l'étude à un (gros) chargement (**upload**), en mode **passif** cette fois, en direction d'un serveur FTP situé en (E) (lisez bien cette phrase)
6. **A** procède aussi au moment de l'étude à un téléchargement de mises à jour par le protocole HTTPS (HTTP+SSL) à partir d'un serveur Web sécurisé d'adresse (F) tournant pour des raisons de sécurité sur le port non standard 8443.
7. Le **serveur web** tournant sur **A** est en connexion avec le serveur MySQL **local** (sur A donc) (imaginez qu'il s'agit d'une plateforme LAMP et que le serveur Web exécute un script PHP qui fait un accès au SGBD **local**, via le réseau loopback donc)
8. Une machine (G) est connectée en **SSH** sur **A**
9. Un utilisateur ayant ouvert une session sur **A** relève son courrier avec le protocole **POP** à partir d'un MDA situé en (H) tandis qu'un autre utilisateur relève son courrier en **IMAP** à partir du même MDA.

1) Pour **chacun des points 1 à 9**, il est demandé de donner **la contribution de ce point** (donc des services, fonctions ou opérations correspondantes) à la table des sockets (**actifs** ou **en écoute**, donc **tous**), TCP et UDP, observable sur la machine **A** (table qui serait le résultat de la commande linux utilisée en TP `#netstat -ntuap`, où l'option "a" veut dire "all", l'option "p" veut dire afficher le nom des processus, l'option "n" veut dire afficher les numéros de ports plutôt que leur nom). On suppose que tous les services utilisent **les ports par défaut**.

Il faut donc remplir et fournir un tableau du type (exactement en fait) :

Point de l'énoncé associé	protocole de transport	adresse locale (socket)	adresse distante (socket)	Etat	Programme à l'origine de la connexion ou du socket en écoute*	Justification**

* imaginer un nom réaliste pour ce programme. Dans les colonnes sockets, vous devez placer des valeurs (de sockets) du type @X:n°port, en inventant au besoin les valeurs nécessaires pour les ports (réalistes et justifiées ; ne mettez pas simplement des croix ou des *, il faut un "vrai" n° de port, réaliste). Pour les adresses IP utiliser la syntaxe @X quand il s'agit de l'adresse d'un des machines **A** à I, et le format X.Y.Z.T sinon.
** ***Mettre une petite phrase d'explication pour chaque ligne du tableau, en justifiant succinctement vos réponses et les valeurs que vous choisissez*** (vous pouvez écrire la justification correspondant à chaque point de l'énoncé en dehors du tableau, en la préfixant du point concerné)

2) Quels sont les ports qu'il faut ouvrir **en entrée** sur le firewall de la machine **A** pour que l'accès aux fonctions et services cités soit possible ?

Exercice 2 : Protocole HTTP *(4 points)*

1) En utilisant **telnet** ou **nc**, quelles seraient les commandes à entrer sur un poste client linux pour récupérer à partir du serveur **truc.ipst.com** et en HTTP/1.0 la ressource `/planning/juin` en précisant qu'on souhaite la version **française** et le format **xhtml** de ce document ? (ce qu'on a fait en TP pour "imiter" un navigateur).

2) Produire aussi la commande "**curl**" correspondante. Vous trouverez en annexe (p14) un extrait du "help" de curl.

3) En utilisant **telnet** ou **nc**, fournir de même les commandes à entrer sur un poste client linux (ou à programmer dans votre code d'un client Web personnalisé) pour envoyer par la méthode **POST** en HTTP/1.1 au formulaire `/commandes/valid.cgi` du serveur web **truc.ipst.com** la valeur (au format "x-www-form-urlencoded") : `article=7045&q=6`

4) Produire aussi la commande "**curl** " correspondante.

5) Quelle serait la syntaxe à utiliser avec telnet/nc si on exploite la méthode **GET** plutôt que POST

6) On suppose que l'URI `/commandes/valid.cgi` est protégée par une authentification HTTP de type "basic". Comment faut-il modifier la commande précédente (expliquez !)

Exercice 3 : DNS *(4 points)*

On réalise sous Linux les 4 requêtes DNS successives (numérotées 1 à 4) présentées page suivante (avec leur réponse)

a) Commencer par dire en 3 lignes **ce qu'on fait globalement** et **ce qu'on obtient** avec ces **quatre** commandes successives. Plus précisément : **que fait chaque commande** et quel est le **lien logique** entre ces 4 requêtes et en quoi consistent **en gros** les questions et les réponses ?

b) Il est ensuite demandé une explication sur chacune des lignes numérotées et **surlignées** (en fait sur la **partie surlignée** de ces lignes) ou sur le bloc de lignes cité. Pour cela **reproduisez** et remplissez le tableau qui suit.

On doit voir que vous comprenez ce que c'est : ce n'est pas vraiment la valeur qui compte mais la **fonction/signification** ; donc **inutile de me recopier la valeur sans expliquer ce qu'elle exprime**. Soyez rigoureux sur les termes que vous utilisez.

Respectez l'ordre des questions posées (donc reproduisez le tableau à l'identique, en remplaçant les questions par vos réponses bien sûr)

ligne(s)	*votre commentaire sur cette ligne ou sa partie surlignée, **en tenant compte des consignes***
1	Les flags ? ça signifie quoi (chaque flag) ? Pourquoi ces valeurs ?
1	Qu'exprime la suite : "QUERY : 1, ANSWER : 1 etc..." ? (tous les champs)
2	1) Donner la signification précise et complète des 4 champs qui suivent le mot SOA 2) A quoi correspond le "164 " ? Ne vous contentez pas d'un sigle. Il faut expliquer ce que c'est et à quoi ça sert 3) Quelle sera cette valeur si je repose la même question dans deux minutes exactement (expliquer, justifier) ? 4) Quelle sera cette valeur si je repose la même question dans deux heures exactement (expliquer, justifier) ?
3	C'est quoi ? ça dit quoi précisément?
4	C'est quoi ? ça dit quoi précisément?
5	C'est quoi ? ça représente quoi ?
6	C'est quoi ? ça dit quoi précisément?
7	C'est quoi ? ça dit quoi précisément?
8	Ça signifie quoi ? Pourquoi ce message ?
9	Les flags ? ça signifie quoi (chaque flag et les message) ? Pourquoi ces valeurs, **en les comparant à ceux de la ligne 1** et ce message (le WARNING) ? (expliquer **en quoi** et **pourquoi** c'est différent de la requête 1 ou 2)
10	C'est quoi ? ça dit quoi précisément (y compris le "300")?
10	Que deviendra la valeur "300" si je repose la même question dans deux minutes ? Et dans 2 heures ? Justifiez !
11	C'est quoi ? ça dit quoi précisément (y compris le "2") ?
12	On voit quoi ici ?
13	à compléter
14	Cette ligne en explique une autre. Laquelle (expliquez)
15	Commentez, expliquez, quel est le problème, est-ce "normal" ?
16	à compléter

```
1 # dig -t SOA amd.com
```

```
  ; <<>> DiG 9.9.4-RedHat-9.9.4-61.el7 <<>> -t SOA amd.com
  ;; global options: +cmd
  ;; Got answer:
  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37358
1 ;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 2

  ;; QUESTION SECTION:
  ;amd.com.                      IN      SOA

  ;; ANSWER SECTION:
2 amd.com.    164     IN      SOA     atl3d.amd.com. hostmaster.atl3d.amd.com. 2019031602 3600 900 3600 3600

  ;; ADDITIONAL SECTION:
3 atl3d.amd.com.        85577   IN      A       165.204.84.65

  ;; Query time: 0 msec
4 ;; SERVER: 10.34.4.1#53(10.34.4.1)
  ;; WHEN: mar. mars 19 17:12:47 CET 2019
  ;; MSG SIZE  rcvd: 105
```

2 # dig -t NS amd.com

```
; <<>> DiG 9.9.4-RedHat-9.9.4-61.el7 <<>> -t NS amd.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36014
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 3

;; QUESTION SECTION:
;amd.com.                       IN      NS

;; ANSWER SECTION:
amd.com.                85565   IN      NS      at13d.amd.com.
amd.com.                85565   IN      NS      cyb3d.amd.com.
amd.com.                85565   IN      NS      sv13d.amd.com.

;; ADDITIONAL SECTION:
at13d.amd.com.          85478   IN      A       165.204.84.65
cyb3d.amd.com.          85478   IN      A       165.204.80.65
sv13d.amd.com.          85478   IN      A       165.204.152.1

;; Query time: 0 msec
;; SERVER: 10.34.4.1#53(10.34.4.1)
;; WHEN: mar. mars 19 17:14:25 CET 2019
;; MSG SIZE  rcvd: 144
```

3 # dig -t ANY amd.com @cyb3d.amd.com. +bufsize=500

```
;; Truncated, retrying in TCP mode.

; <<>> DiG 9.9.4-RedHat-9.9.4-61.el7 <<>> -t ANY amd.com @cyb3d.amd.com. +bufsize=500
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31214
;; flags: qr aa rd; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 3
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;amd.com.                       IN      ANY

;; ANSWER SECTION:
amd.com.                300     IN      SOA     at13d.amd.com. hostmaster.at13d.amd.com. 2019031602 3600
900 3600 3600
amd.com.                7200    IN      NS      sv13d.amd.com.
amd.com.                7200    IN      NS      cyb3d.amd.com.
amd.com.                7200    IN      NS      at13d.amd.com.
amd.com.                300     IN      A       54.225.70.54
amd.com.                300     IN      MX      2 amd-com.mail.protection.outlook.com.
amd.com.                300     IN      TXT
"xKdkpe5EnnNUdvJnYxFz/o1RG/wRdpzXEDyUVPMpSDKwxp2/VEE5kKSyK14wqpUH1NASXFINPAOSImpnXgSSmw=="
amd.com.                300     IN      TXT     "amdext6.and.com."
amd.com.                300     IN      TXT     "amdext2.and.com."
amd.com.                300     IN      TXT     "dropbox-domain-verification=090n3r98zvxg"
amd.com.                300     IN      TXT
"pardot659533-fc50942f2298a3406ad584262d7e87dc047fbf1022a1e0065a8459da6698c86f"
amd.com.                300     IN      TXT     "amdext.and.com."
amd.com.                300     IN      TXT     "amdext5.and.com."

;; ADDITIONAL SECTION:
at13d.amd.com.          300     IN      A       165.204.84.65
sv13d.amd.com.          300     IN      A       165.204.152.1
cyb3d.amd.com.          300     IN      A       165.204.80.65

;; Query time: 275 msec
;; SERVER: : à deviner
;; WHEN: mar. mars 19 17:15:51 CET 2019
;; MSG SIZE  rcvd: 614
```

 # dig -t ANY amd.fr @cyb3d.amd.com.

```
; <<>> DiG 9.9.4-RedHat-9.9.4-61.el7 <<>> -t ANY amd.fr @cyb3d.amd.com.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 43564
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;amd.fr.                        IN      ANY

;; Query time: 265 msec
;; SERVER: : à deviner
;; WHEN: mar. mars 19 17:16:46 CET 2019
```

Exercice 4 : Programmation Sockets *(4 points)*

1)Étudier le code java suivant (fichier "MyProg1.java") et expliquer ce qu'il fait et comment il le fait, en commentant toutes les lignes surlignées.

On fournit plus bas des extraits de la Javadoc concernant 2 classes : *InetAdress* et *Socket* (package java.net), pour compléter les supports de cours ou de TP (qui présentent notamment les classes *DatagramSocket* et *DtatagramPacket*) :

```
1)  import java.net.*;
2)  import java.io.*;
3)
4)  public class MyProg1 {
5)
6)    public static void main(String[] args) {
7)
8)      byte[] buffer = new byte[255];
9)      int port = Integer.parseInt(args[0]);
10)
11)     try {
12)       InetAddress ip=InetAddress.getLocalHost();
13)       System.out.println("Host Name: "+ip.getHostName()+" - IP Address: "+ip.getHostAddress());
14)
15)       try {
16)         DatagramSocket theSocket = new DatagramSocket(port,ip);
17)         while (true) {
18)           DatagramPacket incoming = new DatagramPacket(buffer, buffer.length);
19)           try {
20)             theSocket.receive(incoming);
21)             String s = new String(incoming.getData(), 0, incoming.getLength());
22)             System.out.println(incoming.getAddress() + " at port " + incoming.getPort() + " says : " + s);
23)             DatagramPacket outgoing = new DatagramPacket(incoming.getData(), incoming.getLength(),
    incoming.getAddress(), incoming.getPort());
24)             theSocket.send(outgoing);
25)           }
26)           catch (IOException e) {System.err.println(e);}
27)         }
28)       }
29)       catch (SocketException sockex) {System.err.println(sockex);}
30)     }
31)     catch(Exception e){System.out.println(e);}
32)   }
33) }
```

2) Préciser le "**mode d'emploi**" de cette application pour celui qui l'exécute et celui qui l'exploite. Plus précisément : comment, en partant seulement du code, le testeriez-vous sur une seule machine, un Linux équipé d'un JDK et du logiciel netcat (`nc`) : quelles manips, quelles commandes (raconter : ouvrir un terminal, taper xxx, ouvrir un autre terminal, taper yyy, etc.) ? Que verrait-on à l'écran (dans chaque terminal) ?

3) On lance dans un terminal sur la machine a308-015 (IP 10.34.8.15) l'appli avec la commande : `# java MyProg1 44444` et dans deux autres terminaux deux clients nc qui s'y connectent (un "nc" dans chaque terminal). Que verrait-on en exécutant en même temps dans un autre terminal la commande `# netstat -tuanp | grep 44444` ? Indice : il y a 3 lignes, lesquelles et pourquoi ? Elles ressembleront à quoi ? (mettez dans chaque colonne les valeurs réelles ou inventez-en des réalistes)

4) Modifier maintenant le code de ce programme pour qu'il forwarde le texte reçu, en MAJUSCULE, sur un serveur TCP tournant sur la même machine sur un port fourni en argument (on imagine avoir lancé pour cela un serveur netcat TCP dans un autre terminal sur la même machine). Inspirez-vous du code des clients TCP vus en TP. A défaut d'écrire du "vrai" code Java, vous pouvez faire du pseudo code, en expliquant ce que vous faites à chaque étape. Expliquez aussi ce qu'on fera pour tester ce code et ce qu'on verra sur les différents terminaux utilisés pour ce test.

***

`public class InetAddress`

This class represents an Internet Protocol (IP) address.
An IP address is either a 32-bit or 128-bit unsigned number used by IP, a lower-level protocol on which protocols like UDP and TCP are built.
An instance of an InetAddress consists of an IP address and possibly its corresponding host name (depending on whether it is constructed with a host name or whether it has already done reverse host name resolution).

Modifier and Type	Method and Description (extraits)
`byte[]`	`getAddress()` Returns the raw IP address of this `InetAddress` object.
`static InetAddress`	`getByName(String host)` Determines the IP address of a host, given the host's name.
`String`	`getHostAddress()` Returns the IP address string in textual presentation.
`String`	`getHostName()` Gets the host name for this IP address.
`static InetAddress`	`getLocalHost()` Returns the address of the local host.

public class Socket extends Object implements Closeable

Modifier	Constructor and Description
	Socket() Creates an unconnected socket, with the system-default type of SocketImpl.
	Socket(InetAddress address, int port) Creates a stream socket and connects it to the specified port number at the specified IP address. *Voir plus bas les détails de ce constructeur*
	Socket(InetAddress host, int port, boolean stream) **Deprecated.** *Use DatagramSocket instead for UDP transport.*
	Socket(InetAddress address, int port, InetAddress localAddr, int localPort) Creates a socket and connects it to the specified remote address on the specified remote port.

Modifier and Type	Method and Description
void	**bind**(SocketAddress bindpoint) Binds the socket to a local address.
void	**close**() Closes this socket.
void	**connect**(SocketAddress endpoint) Connects this socket to the server.
InetAddress	**getInetAddress**() Returns the address to which the socket is connected.
InputStream	**getInputStream**() Returns an input stream for this socket.
InetAddress	**getLocalAddress**() Gets the local address to which the socket is bound.
int	**getLocalPort**() Returns the local port number to which this socket is bound.

public Socket(InetAddress address, int port) throws IOException
Creates a stream socket and connects it to the specified port number at the specified IP address.
Parameters:
address - the IP address.
port - the port number.
Throws:
IOException - if an I/O error occurs when creating the socket.
SecurityException - if a security manager exists and its checkConnect method doesn't allow the operation.
IllegalArgumentException - if the port parameter is outside the specified range of valid port values, which is between 0 and 65535, inclusive.
NullPointerException - if address is null.

Exercice 5 : RPC et XDR *(5 points)*

On a développé une application RPC (un service RPC baptisé "imc_calculator") qui retourne, dans sa **première** version, l'Indice de Masse Corporelle (IMC) d'un enfant en fonction de son poids en kilos et de sa taille en centimètres (chiffres ronds), calculé selon la formule IMC = poids/(taille en mètres au carré).
La **deuxième** version de ce programme retourne en plus de l'IMC un commentaire genre "normal" ou "surpoids", en tenant compte aussi de l'âge de l'enfant tel que renseigné, et en consultant des courbes de références.
La **troisième** version fait la même chose en tenant en compte en plus du sexe de l'enfant, et propose aussi une appréciation générale (genre "Tout va bien" ou "Attention") en fonction d'une évolution de cet IMC sur les 3 dernières années (les mesures des 3 dernières années devant bien sûr être transmises pour cela)
Rq : à part le calcul effectif de l'IMC, on ne fera que simuler la génération des commentaires et appréciations (le code retournera toujours la même chose, ce qui ne change rien à l'exercice).

Voici le fichier RPL correspondant (fichier imc.x)

```
struct mesures {
  int poids;
  int taille;
};

typedef struct mesures _mesures;

struct profil {
  _mesures mesures;
  int age;
};
```

```
struct profilMF {
  _profil  profil;
  string sexe<8> ;
};

struct profilMF_glissant {
  string sexe<8>;
  int age;
  _mesures mesures_0;
  _mesures mesures_1;
  _mesures mesures_2;
};

struct resultat {
  float imc;
  string commentaire<100>;
};

program IMC_PROG {
  version IMC_VERSION_1 {
    float IMC(mesures) = 1;
  } = 1;
  version IMC_VERSION_2 {
    struct resultat IMC_COMMENTAIRE(profil) = 1;
  } = 2;
  version IMC_VERSION_3 {
    struct resultat IMC_COMMENTAIRE(profilMF) = 1;
    string IMC_APPRECIATION(profilMF_glissant) = 2;
  } = 3;
} = 0x2000000A;
```

1) Quelles seraient les étapes pour :

a) Produire à partir de ce fichier RPL l'application cliente et l'application serveur (quelles manips ou commandes sur quelles machines avant d'obtenir les exécutables, quels sont les fichiers manipulés ou générés) ? Supposez que vous travaillez sur la machine A308-005

b) Exploiter cette application RPC avec 2 machines Linux distinctes : A308-005 pour le serveur et A308-004 pour le client (quelles manips ou commandes sur quelles machines) ?

2) Sur la machine serveur A308-005 (10.34.8.5) on lance **d'abord** Wireshark (analyseur de protocole) avec filtre RPC pour capturer et étudier les échanges, **à la fois** sur l'interface "loopback" et sur l'interface Ethernet raccordée aux clients. On lance **ensuite** le serveur RPC par `# ./imc_server` (lisez et relisez cette phrase !)
Sur la machine cliente A308-004 (10.34.8.4) on exécute **ensuite** la commande `# rpcinfo -p A308-005`, avec le résultat :

```
   program vers proto   port  service
    100000    4   tcp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    100000    4   udp    111  portmapper
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
 536870922    1   udp    683  imc_calculator
 536870922    2   udp    683  imc_calculator
 536870922    2   tcp    686  imc_calculator
 536870922    3   tcp    686  imc_calculator
```

a) Que signifie le tableau précédent : on voit quoi exactement avec cette commande, que signifie **chaque ligne et chaque colonne de ce tableau** ? Qu'est-ce que ce "portmapper" ? Ne pas recopier les valeurs : ça ne sert à rien ! Il faut **expliquer** à quoi correspond ce tableau, d'où il vient (comment et par quoi il est renseigné), à quoi il sert.

b) À quels "programmes" au sens large un client RPC peut-il prétendre accéder s'il se connecte à distance sur la machine A308-005 ? (dire : il peut accéder à telle chose de telle façon et à telle chose de telle façon et...).

3) On lance **enfin** sur A308-004 le logiciel client, qui teste toutes les versions et toutes procédures RPC du serveur, par la commande :

```
# ./imc_client A308-005  30 120 12 masculin 25 110 20 100
```

Et on obtient sur ce client l'affichage suivant (les réponses du serveur sont en grisé) :

```
Quel est l'IMC de cet enfant de 120 cm et 30 Kg ?
IMC de 20.83

Quel est l'IMC de cet enfant de 12 ans mesurant 120 cm et pesant 30 Kg ?
IMC de 20.83 - Commentaire : Normal pour un enfant de cet âge

Comment évolue l'IMC de cet enfant de sexe masculin de 12 ans mesurant aujourd'hui 120 cm et pesant 30 Kg,
contre 110 cm et 25 Kg et 100 cm et 20 Kg les années précédentes ?
IMC de 20.83 - Commentaire : Normal pour un GARCON de cet âge
Appreciation sur plusieurs années : Sexe masculin - Age 12 - (100 cm,20 Kg) puis (110 cm,25 Kg) puis (120 cm,30
Kg) --> Tout va bien
```

On s'intéresse alors à ce qui a été capturé sur le serveur par wireshark depuis le début (la question 2) : la totalité de l'échange, plus des "zooms" sur les trames 11, 16, 19, 24, 29 :

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	Portmap	100	V2 UNSET Call (Reply In 2)
2	0.000222	127.0.0.1	127.0.0.1	Portmap	72	V2 UNSET Reply (Call In 1)
3	0.000464	127.0.0.1	127.0.0.1	Portmap	100	V2 UNSET Call (Reply In 4)
4	0.000590	127.0.0.1	127.0.0.1	Portmap	72	V2 UNSET Reply (Call In 3)
5	0.000755	127.0.0.1	127.0.0.1	Portmap	100	V2 UNSET Call (Reply In 6)
6	0.000877	127.0.0.1	127.0.0.1	Portmap	72	V2 UNSET Reply (Call In 5)
7	0.001138	127.0.0.1	127.0.0.1	Portmap	100	V2 SET Call (Reply In 8)
8	0.001271	127.0.0.1	127.0.0.1	Portmap	72	V2 SET Reply (Call In 7)
9	0.001461	127.0.0.1	127.0.0.1	Portmap	100	V2 SET Call (Reply In 10)
10	0.001566	127.0.0.1	127.0.0.1	Portmap	72	V2 SET Reply (Call In 9)
11	0.001755	127.0.0.1	127.0.0.1	Portmap	100	V2 SET Call (Reply In 12)
12	0.001892	127.0.0.1	127.0.0.1	Portmap	72	V2 SET Reply (Call In 11)
13	0.002084	127.0.0.1	127.0.0.1	Portmap	100	V2 SET Call (Reply In 14)
14	0.002205	127.0.0.1	127.0.0.1	Portmap	72	V2 SET Reply (Call In 13)
15	9.068465	10.34.8.4	10.34.8.5	Portmap	112	V2 DUMP Call (Reply In 16)
16	9.068752	10.34.8.5	10.34.8.4	Portmap	300	V2 DUMP Reply (Call In 15)
17	19.868837	10.34.8.4	10.34.8.5	Portmap	100	V2 GETPORT Call (Reply In 18) RPC:536870922(536870922) V:1 UDP
18	19.869079	10.34.8.5	10.34.8.4	Portmap	72	V2 GETPORT Reply (Call In 17) Port:688
19	19.869352	10.34.8.4	10.34.8.5	RPC:536870922	92	V1 proc-1 Call (Reply In 20)
20	19.869449	10.34.8.5	10.34.8.4	RPC:536870922	72	V1 proc-1 Reply (Call In 19)
21	19.870420	10.34.8.4	10.34.8.5	Portmap	100	V2 GETPORT Call (Reply In 22) RPC:536870922(536870922) V:2 UDP
22	19.870652	10.34.8.5	10.34.8.4	Portmap	72	V2 GETPORT Reply (Call In 21) Port:688
23	19.870861	10.34.8.4	10.34.8.5	RPC:536870922	96	V2 proc-1 Call (Reply In 24)
24	19.870936	10.34.8.5	10.34.8.4	RPC:536870922	112	V2 proc-1 Reply (Call In 23)
25	19.871621	10.34.8.4	10.34.8.5	Portmap	100	V2 GETPORT Call (Reply In 26) RPC:536870922(536870922) V:3 UDP
26	19.871799	10.34.8.5	10.34.8.4	Portmap	72	V2 GETPORT Reply (Call In 25) Port:688
27	19.872021	10.34.8.4	10.34.8.5	RPC:536870922	108	V3 proc-1 Call (Reply In 28)
28	19.872097	10.34.8.5	10.34.8.4	RPC:536870922	112	V3 proc-1 Reply (Call In 27)
29	19.872328	10.34.8.4	10.34.8.5	RPC:536870922	124	V3 proc-2 Call (Reply In 30)
30	19.872414	10.34.8.5	10.34.8.4	RPC:536870922	172	V3 proc-2 Reply (Call In 29)

```
⊞ Frame 11: 100 bytes on wire (800 bits), 100 bytes captured (800 bits)
⊞ Linux cooked capture
⊞ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
⊞ User Datagram Protocol, Src Port: asipregistry (687), Dst Port: sunrpc (111)
⊟ Remote Procedure Call, Type:Call XID:0x214436b3
    XID: 0x214436b3 (558118579)
    Message Type: Call (0)
    RPC Version: 2
    Program: Portmap (100000)
    Program Version: 2
    Procedure: SET (1)
    [The reply to this request is in frame 12]
  ⊞ Credentials
  ⊞ Verifier
⊟ Portmap
    [Program Version: 2]
    [V2 Procedure: SET (1)]
    Program: RPC:536870922 (536870922)
    Version: 2
    Proto: TCP (6)
    Port: 686
```

```
⊞ Frame 16: 300 bytes on wire (2400 bits), 300 bytes captured (2400 bits)
⊞ Linux cooked capture
⊞ Internet Protocol Version 4, Src: 10.34.8.5 (10.34.8.5), Dst: 10.34.8.4 (10.34.8.4)
⊞ Transmission Control Protocol, Src Port: sunrpc (111), Dst Port: 757 (757), Seq: 1, Ack: 45, Len: 232
⊟ Remote Procedure Call, Type:Reply XID:0x3d13b5ea
  ⊞ Fragment header: Last fragment, 228 bytes
    XID: 0x3d13b5ea (1024701930)
    Message Type: Reply (1)
    [Program: Portmap (100000)]
    [Program Version: 2]
    [Procedure: DUMP (4)]
    Reply State: accepted (0)
    [This is a reply to a request in frame 15]
    [Time from request: 0.000287000 seconds]
  ⊞ Verifier
    Accept State: RPC executed successfully (0)
⊟ Portmap
    [Program Version: 2]
    [V2 Procedure: DUMP (4)]
    Value Follows: Yes
  ⊞ Map Entry: Portmap (100000) V4
    Value Follows: Yes
  ⊞ Map Entry: Portmap (100000) V3
    Value Follows: Yes
  ⊞ Map Entry: Portmap (100000) V2
    Value Follows: Yes
  ⊞ Map Entry: Portmap (100000) V4
    Value Follows: Yes
  ⊞ Map Entry: Portmap (100000) V3
    Value Follows: Yes
  ⊞ Map Entry: Portmap (100000) V2
    Value Follows: Yes
  ⊞ Map Entry: RPC:536870922 (536870922) V1
    Value Follows: Yes
  ⊞ Map Entry: RPC:536870922 (536870922) V2
    Value Follows: Yes
  ⊞ Map Entry: RPC:536870922 (536870922) V2
    Value Follows: Yes
  ⊞ Map Entry: RPC:536870922 (536870922) V3
    Value Follows: No
```

```
⊞ Frame 19: 92 bytes on wire (736 bits), 92 bytes captured (736 bits)
⊞ Linux cooked capture
⊞ Internet Protocol Version 4, Src: 10.34.8.4 (10.34.8.4), Dst: 10.34.8.5 (10.34.8.5)
⊞ User Datagram Protocol, Src Port: 759 (759), Dst Port: corba-iiop (683)
⊟ Remote Procedure Call, Type:Call XID:0x4c2f201f
    XID: 0x4c2f201f (1278156831)
    Message Type: Call (0)
    RPC Version: 2
    Program: RPC:536870922 (536870922)
    Program Version: 1
    Procedure: proc-1 (1)
    [The reply to this request is in frame 20]
  ⊞ Credentials
  ⊞ Verifier
⊟ Unknown RPC Program:536870922
    [Program Version: 1]
    [Procedure: 1]
  ⊟ Data (8 bytes)
      Data: 0000001e00000078
      [Length: 8]
```

```
0000  00 00 00 01 00 06 38 63  bb 8a 50 ef 00 00 08 00   ......8c ..P.....
0010  45 00 00 4c 00 00 40 00  40 11 16 35 0a 22 08 04   E..L..@. @..5."..
0020  0a 22 08 05 02 f7 02 ab  00 38 48 9d 4c 2f 20 1f   ."...... .8H.L/ .
0030  00 00 00 00 00 00 00 02  20 00 00 0a 00 00 00 01   ........  .......
0040  00 00 00 01 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0050  00 00 00 00 00 00 00 1e  00 00 00 78               ........ ....
```

```
⊞ Frame 24: 112 bytes on wire (896 bits), 112 bytes captured (896 bits)
⊞ Linux cooked capture
⊞ Internet Protocol Version 4, Src: 10.34.8.5 (10.34.8.5), Dst: 10.34.8.4 (10.34.8.4)
⊞ User Datagram Protocol, Src Port: corba-iiop (683), Dst Port: 761 (761)
⊟ Remote Procedure Call, Type:Reply XID:0x34e08553
    XID: 0x34e08553 (887129427)
    Message Type: Reply (1)
    [Program: RPC:536870922 (536870922)]
    [Program Version: 2]
    [Procedure: proc-1 (1)]
    Reply State: accepted (0)
    [This is a reply to a request in frame 23]
    [Time from request: 0.000071000 seconds]
  ⊞ Verifier
    Accept State: RPC executed successfully (0)
⊟ Unknown RPC Program:536870922
    [Program Version: 2]
    [Procedure: proc-1 (1)]
  ⊟ Data (44 bytes)
      Data: 41abaaab000000214e6f726d616c20706f7572207568e2065...
      [Length: 44]
```

```
0000  00 04 00 01 00 06 30 8d  99 25 b3 fc 00 00 08 00   ......0. .%......
0010  45 00 00 60 00 00 40 00  40 11 16 41 0a 22 08 05   E..`..@. @..A."..
0020  0a 22 08 04 02 ab 02 f9  00 4c 24 aa 34 e0 85 53   ."...... .L$.4..S
0030  00 00 00 01 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0040  00 00 00 00 41 ab aa ab  00 00 00 21 4e 6f 72 60   ....A... ...!Nor
0050  61 6c 20 70 6f 75 72 20  75 6e 20 65 6e 66 61 6e   al pour  un enfan
0060  74 20 84 65 20 63 65 74  20 c3 a2 67 65 00 00 00   t de cet  ...ge...
```

```
⊞ Frame 29: 124 bytes on wire (992 bits), 124 bytes captured (992 bits)
⊞ Linux cooked capture
⊞ Internet Protocol Version 4, Src: 10.34.8.4 (10.34.8.4), Dst: 10.34.8.5 (10.34.8.5)
⊞ User Datagram Protocol, Src Port: 763 (763), Dst Port: corba-iiop (683)
⊟ Remote Procedure Call, Type:Call XID:0x78b7dfb9
    XID: 0x78b7dfb9 (2025316281)
    Message Type: Call (0)
    RPC Version: 2
    Program: RPC:536870922 (536870922)
    Program Version: 3
    Procedure: proc-2 (2)
    [The reply to this request is in frame 30]
  ⊞ Credentials
  ⊞ Verifier
⊟ Unknown RPC Program:536870922
    [Program Version: 3]
    [Procedure: proc-2 (2)]
  ⊟ Data (40 bytes)
      Data: 0000000386d1736375bc096e0000000c0000001e00000078...
      [Length: 40]
```

```
0000  00 00 00 01 00 06 38 63  bb 8a 50 ef 00 00 08 00   ......8c ..P.....
0010  45 00 00 6c 00 00 40 00  40 11 16 35 0a 22 08 04   E..l..@. @..5."..
0020  0a 22 08 05 02 fb 02 ab  00 58 9b 80 78 b7 df b9   ."...... .X..x...
0030  00 00 00 00 00 00 00 02  20 00 00 0a 00 00 00 03   ........  .......
0040  00 00 00 02 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0050  00 00 00 00 00 00 00 08  00 01 73 63 75 6c 69 6e   ........ ..masculin
0060  00 00 00 0c 00 00 00 1e  00 00 00 78 00 00 00 14   ........ ...x....
0070  00 00 00 6e 00 00 00 14  00 00 00 64               ...n.... ...d
```

a) En vous basant sur les champs "adresses", le champ "Info", la valeur des ports dans les "zooms" et surtout sur la logique d'exécution des programmes RPC, dérouler à partir de cette capture Wireshark le scénario (que se passe-t-il, étape par étape, sans forcément détailler trame par trame mais sans en oublier (vous pouvez en regrouper plusieurs) ; qui fait quoi à chaque étape et surtout pourquoi) ?

b) Que trouve-t-on dans la trame 11 et pourquoi ? (il se passe quoi, pourquoi c'est là, ça sert à quoi ?)

c) Même question pour la 16

d) Décoder (donc retrouvez, en donnant les "calculs" ou raisonnements réalisés par vous) et justifiez (pourquoi c'est là et pourquoi ça vaut ça) les données (leur sens et leur valeur) transportées par la trame 19 (champ "Data"). C'est à mettre en relation avec l'affichage sur le client p7

e) Même question pour la 24. Identifiez au moins la structure de la réponse, genre : "un entier sur 8 octets", "une structure sur 12 octets contenant etc..."

f) Même question pour la 29

Rq : vous disposez plus loin d'un extrait du RFC4506

Voici (à titre informatif) le code C du Fichier "serveur" `imc_server.c` : ***(Vous n'en avez pas vraiment besoin !)***

```
#include "imc.h"

float * imc_1_svc(mesures *argp, struct svc_req *rqstp)
{
  static float  result;
  result = 10000 *  (float) argp ->poids / ((float) argp => taille * (float) argp -> taille);
  return &result;
}

struct resultat * imc_commentaire_2_svc(profil *argp, struct svc_req *rqstp)
{
  static struct resultat  result;
  result.imc=10000 *  (float) argp =>mesures.poids / ((float) argp -> mesures.taille * (float) argp -> mesures.taille) ;
/* là on imagine que l'on compare cet IMC à des abaques qui tiennent compte de l'âge de l'enfant et on en tire un commentaire*/
  result.commentaire = "Normal pour un enfant de cet âge";
  return &result;
}

struct resultat * imc_commentaire_3_svc(profilMF *argp, struct svc_req *rqstp)
{
  static struct resultat  result;
  result.imc=10000 *  (float) argp ->profil.mesures.poids / ((float) argp -> profil.mesures.taille * (float) argp =>
profil.mesures.taille) ;
/* là on imagine que l'on compare cet IMC à des abaques qui tiennent compte de l'âge et du sexe de l'enfant et on en tire un
commentaire*/
  if (strcmp(argp->sexe,"masculin") ==0) result.commentaire = "Normal pour un GARCON de cet âge";
  else if (strcmp(argp->sexe,"feminin") ==0) result.commentaire = "Normal pour un FILLE de cet âge";
  else result.commentaire = "Veuillez préciser le sexe de l'enfant : masculin ou feminin";
  return &result;
}

char ** imc_appreciation_3_svc(profilMF_glissant *argp, struct svc_req *rqstp)
{
  static char * result;
/* là on imagine que l'on regarde la taille et le poids sur 3 ans et que l'on compare ces nombres à des abaques
qui tiennent compte de l'âge et du sexe de l'enfant. Et on en tire une appreciation générale */
  char *str;
  asprintf(&str,"Sexe %s - Age %i - (%i cm,%i Kg) puis (%i cm,%i Kg) puis (%i cm,%i Kg) --> Tout va bien\n",argp->sexe, argp-
>age,(argp->mesures_2).taille, (argp->mesures_2).poids,(argp->mesures_1).taille,(argp->mesures_1).poids,(argp-
>mesures_0).taille, (argp->mesures_0).poids);
  result= str;
  return &result;
}
```

Voici (à titre informatif) le code C du fichier "client" : `imc_client.c` : ***(Vous n'en avez pas vraiment besoin !)***

```
#include "imc.h"

void imc_prog_1(char *host, int poids, int taille)
{
  CLIENT *clnt;
  float  *result_1;
  mesures  imc_1_arg;

#ifndef DEBUG
  clnt = clnt_create (host, IMC_PROG, IMC_VERSION_1, "udp");
  if (clnt == NULL) {
    clnt_pcreateerror (host);
    exit (1);
  }
#endif/* DEBUG */

  imc_1_arg.poids=poids;imc_1_arg.taille=taille;
  result_1 = imc_1(&imc_1_arg, clnt);
  if (result_1 == (float *) NULL) {
    clnt_perror (clnt, "call failed");
  }
  else printf ("IMC ou %7.2f\n",(float) *result_1);
#ifndef DEBUG
  clnt_destroy (clnt);
```

```
}

void imc_prog_2(char *host, int poids, int taille,int age)
{
  CLIENT *clnt;
  struct resultat  *result_1;
  profil  imc_commentaire_2_arg;

#ifndef DEBUG
  clnt = clnt_create (host, IMC_PROG, IMC_VERSION_2, "udp");
  if (clnt == NULL) {
    clnt_pcreateerror (host);
    exit (1);
  }
#endif/* DEBUG */

  imc_commentaire_2_arg.mesures.poids=poidsimc_commentaire_2_arg.mesures.taille=taille;imc_commentaire_2_arg.age=age;
  result_1 = imc_commentaire_2(&imc_commentaire_2_arg, clnt);
  if (result_1 == (struct resultat *) NULL) {
    clnt_perror (clnt, "call failed");
  }
  else printf ("IMC de %2.2f - Commentaire : %s\n", (float) (*result_1).imc, (char *) (*result_1).commentaire);
#ifndef DEBUG
  clnt_destroy (clnt);
#endif /* DEBUG */
}
void imc_prog_3(char *host, int poids_0, int taille_0, int age, char *sexe, int poids_1, int taille_1, int poids_2,  int
taille_2)

{
  CLIENT *clnt;
  struct resultat  *result_1;
  profilIMC  imc_commentaire_3_arg;
  char * *result_2;
  profilIMC_qlissant  imc_appreciation_3_arg;

#ifndef DEBUG
  clnt = clnt_create (host, IMC_PROG, IMC_VERSION_3, "udp");
  if (clnt == NULL) {
    clnt_pcreateerror (host);
    exit (1);
  }
#endif/* DEBUG */

  imc_commentaire_3_arg.profil.mesures.poids=poids_0;imc_commentaire_3_arg.profil.mesures.taille=taille_0;
  imc_commentaire_3_arg.sexe=sexe;
  result_1 = imc_commentaire_3(&imc_commentaire_3_arg, clnt);
  if (result_1 == (struct resultat *) NULL) {
    clnt_perror (clnt, "call failed");
  }
  else printf ("IMC de %2.2f - Commentaire : %s\n", (float) (*result_1).imc, (char *) (*result_1).commentaire);

  imc_appreciation_3_arg.sexe = sexe; imc_appreciation_3_arg.age=age;
  imc_appreciation_3_arg.mesures_0.poids = poids_0;imc_appreciation_3_arg.mesures_0.taille=taille_0;
  imc_appreciation_3_arg.mesures_1.poids = poids_1;imc_appreciation_3_arg.mesures_1.taille=taille_1;
  imc_appreciation_3_arg.mesures_2.poids = poids_2;imc_appreciation_3_arg.mesures_2.taille=taille_2;
  result_2 = imc_appreciation_3(&imc_appreciation_3_arg, clnt);
  if (result_2 == (char **) NULL) {
    clnt_perror (clnt, "call failed");
  }
  else printf ("Appreciation sur plusieurs années : %s\n", *result_2);
#ifndef DEBUG
  clnt_destroy (clnt);
#endif /* DEBUG */
}

int main (int argc, char *argv[])
{
  char *host;

  if (argc < 2) {
    printf ("usage: %s server_host\n", argv[0]);
    exit (1);
  }
  host = argv[1];
  int poids=atoi(argv[2]);int taille=atoi(argv[3]);int age=atoi(argv[4]);char *sexe=argv[5];

  printf("\nQuel est l'IMC de cet enfant de %i cm et %i Kg ?\n", taille,poids);
  imc_prog_1 (host,poids,taille);

  printf("\nQuel est l'IMC de cet enfant de %i ans mesurant %i cm et pesant %i Kg ?\n",age, taille,poids);
  imc_prog_2 (host,poids,taille,age);

  int taille_1=atoi(argv[7]),poids_1=atoi(argv[6]), taille_2=atoi(argv[9]),poids_2=atoi(argv[8]);
  printf("\nComment évolue l'IMC de cet enfant de sexe %s de %i ans mesurant aujourd'hui %i cm et pesant %i Kg , contre %i cm et
%i Kg et %i cm et %i Kg les années précédentes ?\n",sexe, age, taille,poids, taille_1,poids_1, taille_2,poids_2);
  imc_prog_3 (host,poids,taille,age,sexe,poids_1,taille_1,poids_2,taille_2);

exit (0);
}
```

Extraits du RFC4506 :

4.2. Unsigned Integer

An XDR unsigned integer is a 32-bit datum that encodes a non-negative integer in the range [0,4294967295]. It is represented by an unsigned binary number whose most and least significant bytes are 0 and 3, respectively. An unsigned integer is declared as follows:

```
      unsigned int identifier;
```

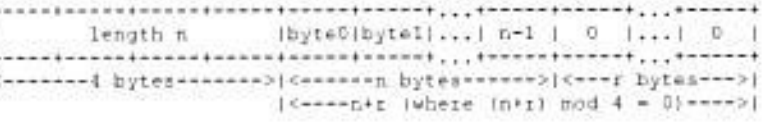
4.11. String

The standard defines a string of n (numbered 0 through n-1) ASCII bytes to be the number n encoded as an unsigned integer (as described above), and followed by the n bytes of the string. Byte n of the string always precedes byte n+1 of the string, and byte 0 of the string always follows the string's length. If n is not a multiple of four, then the n bytes are followed by enough (0 to 3) residual zero bytes, r, to make the total byte count a multiple of four. Counted byte strings are declared as follows:

```
      string object<m>;       or          string object<>;
```

The constant m denotes an upper bound of the number of bytes that a string may contain. If m is not specified, as in the second declaration, it is assumed to be (2**32) - 1, the maximum length. The constant m would normally be found in a protocol specification. For example, a filing protocol may state that a file name can be no longer than 255 bytes, as follows:

```
      string filename<255>;
```



4.6. Floating-Point

The standard defines the floating-point data type "float" (32 bits or 4 bytes). The encoding used is the IEEE standard for normalized single-precision floating-point numbers [IEEE]. The following three fields describe the single-precision floating-point number:

S: The sign of the number. Values 0 and 1 represent positive and negative, respectively. One bit.

E: The exponent of the number, base 2. 8 bits are devoted to this field. The exponent is biased by 127.

F: The fractional part of the number's mantissa, base 2. 23 bits are devoted to this field.

Therefore, the floating-point number is described by:

```
      (-1)**S * 2**(E-Bias) * 1.F
```

It is declared as follows:

```
      float identifier;
```

Just as the most and least significant bytes of a number are 0 and 3, the most and least significant bits of a single-precision floating-point number are 0 and 31. The beginning bit (and most significant bit) offsets of S, E, and F are 0, 1, and 9, respectively. Note that these numbers refer to the mathematical positions of the bits, and NOT to their actual physical locations (which vary from medium to medium).

4.14. Structure

Structures are declared as follows:

```
      struct {
         component-declaration-A;
         component-declaration-B;
         ...
      } identifier;
```

The components of the structure are encoded in the order of their declaration in the structure. Each component's size is a multiple of four bytes, though the components may be different sizes.

Exercice 6 : MAIL et format MIME *(4 points)*

Regardez la sortie écran qui suit (les commandes entrées par l'utilisateur et les réponses ; les n° de lignes ont été ajoutées après coup) :

```
1)  nc  mail.ens.cnam-mp.fr 110
2)  +OK Email service Dovecot (pop3,imap) ready - ipst-info.net.  NE PAS EFFACER LES 5 PREMIERS MESSAGES !
3)  USER tp33
4)  +OK
5)  PASS pwd33
6)  +OK logged in.
7)  LIST
8)  +OK 5 messages:
9)  1 1651
10) 2 2478
11) 3 2417
12) 4 31561
13) 5 12335
14) .
15) RETR 4
16) +OK 31561 octets
17) Return-Path: <tp33@ipst-info.net>
18) Delivered-To: tp33@ipst-info.net
19) Received: from ipst-info.net (vpn.ipst-info.net [172.16.0.10])
20)  by mail.ipst-info.net (Postfix) with ESMTP id 93BD61FEC5;
21)  Mon, 13 Mar 2017 07:38:50 +0100 (CET)
22) Received: from s321-002.ens.cnam-mp.fr ([10.34.7.2]) by ipst-info.net with Microsoft SMTPSVC(7.5.7601.17514);
23)   Mon, 13 Mar 2017 15:45:06 +0100
24) Message-ID: <58CC8A71.70207038@ipst-info.net>
25) Date: Mon, 13 Mar 2017 15:45:05 +0100
26) From: tp30 <tp30@ipst-info.net>
27) User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:10.0.12) Gecko/20130108 Thunderbird/10.0.12
28) MIME-Version: 1.0
29) To: tp33@ipst-info.net, tp34@ipst-info.net
30) Subject: Message avec un long sujet =?ISO-8859-15?Q?int=E9grant_quelque?=
31)  =?ISO-8859-15?Q?s_lettres_accentu=E9es_et_le_symbole_mon=E9t?=
32)  =?ISO-8859-15?Q?aire_=A4_plus_une_pi=E8ce_jointe_et_un_s?=
33)  =?ISO-8859-15?Q?miley?=
34) Content-Type: multipart/mixed;
35)  boundary="------------080306070409010703040500"
36) X-OriginalArrivalTime: 13 Mar 2017 14:45:06.0220 (UTC) FILETIME=[47CB16C0:01D29C08]
37)
38) This is a multi-part message in MIME format.
39) --------------080306070409010703040500
40) Content-Type: multipart/alternative;
41)  boundary="------------050105010803030202000303"
42)
43) --------------050105010803030202000303
44) Content-Type: text/plain; charset=ISO-8859-15; format=flowed
45) Content-Transfer-Encoding: 8bit
46)
47) Heureux d'être le b- officiaire de cette somme de 100 €
48) Ci-joint mon CV.
49) SOURIRE
50)
51) --------------050105010803030202000303
52) Content-Type: multipart/related;
53)  boundary="------------020601080605050408040104"
54)
55) --------------020601080605050408040104
56) Content-Type: text/html; charset=ISO-8859-15
57) Content-Transfer-Encoding: 8bit
58)
59) <html>
60)   <head>
61)     <meta content="text/html; charset=ISO-8859-15"
62)       http-equiv="Content-Type">
63)   </head>
64)   <body text="#000000" bgcolor="#FFFFFF">
65)     Heureux d'être le b- officiaire de cette somme de 100 €<br>
66)     Ci-joint mon CV.<br>
67)     <img alt="SOURIRE" src="cid:part1.09070100.01060506@ipst-info.net"
68)       height="16" width="16"><br>
69)   </body>
70) </html>
71)
72) --------------020601080605050408040104
73) Content-Type: image/png;
74)  name="emoticon-30-16.png"
75) Content-Transfer-Encoding: base64
76) Content-ID: <part1.09070100.01060506@ipst-info.net>
77) Content-Disposition: inline;
78)  filename="emoticon-30-16.png"
79)
80) iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAYAAAAf8/9hAAAABmJLR0QA/wD/AP+gvaeTAAAA
81) ...
82) YzkGMIv+7HeP/zr/qIw5j6mL19+255rQv6pGNwy7/hkM421S33MEWkJr9V/d2v2oh23kgbEk
83) b9c1fwCYaKmS1fK91gAAAABJRU5ErkJggg==
84) --------------020601080605050408040104--
85)
86) --------------050105010803030202000303--
87)
88) --------------080306070409010703040500
89) Content-Type: application/pdf;
90)  name="cv.pdf"
91) Content-Transfer-Encoding: base64
92) Content-Disposition: attachment;
93)  filename="cv.pdf"
94)
95) JVBERi0xLjY%Ze1je9MNCjM3IDAgb2JqIDw8ICAgbmVhcmlfZWQgMS9MIDIwNTk3LDE0qNDAw
96) RIAANDEwNS9OIDEvVCAwJTc5OSIIIEwgNTAwNIAyMTVdPj4NIW5kb2JqDQAgICAgICAgICAgICAg
97) ICAgICAgDQp4cmVmDQowIyAzNCINCjAwMDAwMDAwMDAwNiAwMDAwMCBuDQowMDAwMDAzMzg2IDAw
98) .. [oi on a coupé]
99) IDFINTMIIGchCjAwMDAwMDAwMDAgNjU1MzUgZgINCjAwMDAwMDAwMDAgMDAzNTUzNSBnDQowMDAw
100) MDAwMDAwIDYINTKIIGYNCiAyMa1sIKINCjw8LlNpemUgMzcvRW5jcnlwdCAzOCAwIFI+PgCH
101) o3RhcnR4cmVmDQoxMTYNCiUlRU0QDQo=
102) --------------080306070409010703040500--
```

1) Sans s'intéresser pour l'instant au détail du mail récupéré, expliquer ce qu'on fait et ce qui se passe à chaque étape/commande (**de la ligne 1 jusqu'à la ligne 16**). Dire au moins : "on fait ça, et il se passe ça". Vous pouvez utiliser les n° de ligne pour désigner une étape.

2) Que signifie la ligne 27, qu'apprend-on ?

3) Quel est le sujet/l'objet de ce mail ? (dire ce qu'on verra exactement dans la ligne "objet" dans un MUA type Outlook ou Thunderbird). **Expliquer** bien sûr ce qui vous permet de répondre et donc **quelle est la technique utilisée** pour mettre en forme ce sujet. Autrement dit : donner l'algorithme précis et complet qui permet à un MUA de "décoder" ces données. Ne me donnez pas le "sujet" sans les explications techniques : ça n'aura aucune valeur !

4) Quelle est la **structure** de ce message ? Faire un schéma (genre ci-dessous) avec **la hiérarchie des blocs** et **le type MIME de chacun** de ces blocs. **Expliquer** ce qui vous permet de "reconstruire" cette structure. Autrement dit : donner l'algorithme précis et complet qui permet à un MUA de reconstruire et d'afficher correctement le message.

5) Quel sera l'affichage produit par un client de messagerie graphique type Outlook ou Thunderbird ; il faut dire : "on verra ça, puis ça, et/ou ça" etc... (exactement !)

6) Pourquoi voit-on ces caractères bizarres à l'écran (**◆**) ? Quel est le problème (et sa solution) ?

7) Commentez/expliquez les lignes 49, 67, 76 (ensemble)

8) Commentez/expliquez **chacune** des lignes 89 à 93 (et à l'intérieur des lignes **chacun des mots ou groupes de mots**, et notamment **le "base64"**) : qu'est-ce que ça veut dire, pourquoi c'est là, à quoi ça sert ? Autrement dit : donner l'algorithme précis et complet qui permet à un MUA de reconstruire et d'afficher correctement cette partie du message.

Exercice 7 : Web Services XML *(4 points)*

Soit un service Web XML situé à l'adresse :
http://a308-012.ens.cnam-mp.fr:8080/fibowebapp/fibonacci
Ce service renvoie, quand on lui fournit un entier N, la valeur du $N^{ème}$ élément de la suite de Fiboracci (une suite d'entiers dans laquelle chaque terme est la somme des deux termes qui le précèdent ; mais peu importe)

1) Quel est la place et le rôle du **WSDL** dans la mise en œuvre d'un service Web ? Comparer cette place et ce rôle à ce que vous connaissez des **RPC**

2) Quelles sont les **étapes et commandes** à exécuter avant de pouvoir développer et mettre en œuvre un programme client capable d'exploiter ce service Web XML

   a. en environnement .NET (vous avez un SDK .NET et une console DOS)
   b. en environnement java (vous avez un JDK et un terminal)

Plus que les commandes exactes, il faut **expliquer** pourquoi on procède de la sorte et qu'est ce qui se passe alors.

3) Voici le document qui correspond au WS. Quel est son **contenu** approximatif (en quelques lignes) : on trouve quoi dedans, qu'est-ce que ça dit, quel est le lien logique entre les différentes parties ?

```
<?xml version='1.0' encoding='UTF-8'?>
<definitions
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://ws.fibowebapp.dvp.com/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://ws.fibowebapp.dvp.com/"
    name="FibonacciWebServiceService">
    <types />
    <message name="fibonacci">
        <part name="rang" type="xsd:int" />
    </message>
    <message name="fibonacciResponse">
        <part name="return" type="xsd:long" />
    </message>
```

```
    <portType name="FibonacciWebService">
        <operation name="fibonacci">
            <input
                wsam:Action="http://ws.fibowebapp.dvp.com/FibonacciWebService/fibonacciRequest"
                message="tns:fibonacci" />
            <output
                wsam:Action="http://ws.fibowebapp.dvp.com/FibonacciWebService/fibonacciResponse"
                message="tns:fibonacciResponse" />
        </operation>
    </portType>
    <binding name="FibonacciWebServicePortBinding" type="tns:FibonacciWebService">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
            style="rpc" />
        <operation name="fibonacci">
            <soap:operation soapAction="" />
            <input>
                <soap:body use="literal" namespace="http://ws.fibowebapp.dvp.com/" />
            </input>
            <output>
                <soap:body use="literal" namespace="http://ws.fibowebapp.dvp.com/" />
            </output>
        </operation>
    </binding>
    <service name="FibonacciWebServiceService">
        <port name="FibonacciWebServicePort" binding="tns:FibonacciWebServicePortBinding">
            <soap:address location="http://a308-012.ens.cnam-np.fr:8080/fibowebapp/fibonacci" />
        </port>
    </service>
</definitions>
```

4) Commentez les captures suivantes (contenu « utile » des paquets qui circulent entre le client qui consomme le service Web et le serveur qui le fournit). Il est demandé d'expliquer chaque ligne ou bloc de ligne, **en faisant le lien avec le WSDL précédent**. Expliquez aussi ce qu'est par exemple le « ws » ou le « ns2 » qu'on y trouve.
Quel est la valeur et le "type" (comme quand vous déclarez des variables dans un code) des données qui circulent ?

Message SOAP de la requête :

```
<soapenv:Envelope
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:ws="http://ws.fibowebapp.dvp.com/">
 <soapenv:Header/>
 <soapenv:Body>
    <ws:fibonacci>
       <rang>14</rang>
    </ws:fibonacci>
 </soapenv:Body>
</soapenv:Envelope>
```

Message SOAP de la réponse :

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
    <ns2:fibonacciResponse xmlns:ns2="http://ws.fibowebapp.dvp.com/">
       <return>377</return>
    </ns2:fibonacciResponse>
 </S:Body>
</S:Envelope>
```

Annexe : extrait de "curl -help" (pour exercice 2)

```
Usage: curl [options...] <url>
 -d, --data DATA     HTTP POST data (H)
 -G, --get           Send the -d data with a HTTP GET (H)
 -H, --header LINE   Custom header to pass to server (H)
 -I, --head          Show document info only
 -0, --http1.0       Use HTTP 1.0 (H)
 -i, --include       Include protocol headers in the output (H/F)
 -k, --insecure      Allow connections to SSL sites without certs (H)
 -Q, --quote CMD     Send command(s) to server before transfer (F/SFTP)
 -X, --request COMMAND  Specify request command to use
 -T, --upload-file FILE  Transfer FILE to destination
 -B, --use-ascii     Use ASCII/text transfer
 -u, --user USER[:PASSWORD]  Server user and password
 -A, --user-agent STRING  User-Agent to send to server (H)
 -v, --verbose       Make the operation more talkative
```