Euro graduation access
Concours eg@

2015

Consortium International

Epreuve d'INFORMATIQUE

Informations sur l'épreuve

Barème :	/20
Durée :	90 min
Calculatrice autorisée :	Non

Merci de ne rien marquer sur le sujet.
Pour chaque question de l'épreuve, veuillez choisir la (les) bonne(s) réponse(s).
Répondez sur la grille de réponses séparée.
Uniquement les grilles de réponses correctement remplies seront corrigées.

Partie 1 : Questions générales

1. **Lorsqu'une adresse électronique apparait dans le champ CC d'un courriel, quel est son statut ?**
   A. C'est l'adresse du destinataire qui est mis en copie de ce courriel secrètement, sans que les autres destinataires le sachent
   B. C'est l'adresse de l'expéditeur du courriel
   C. C'est l'adresse du destinataire principal, à qui le courriel s'adresse directement.
   D. C'est l'adresse du destinataire qui est mis en copie de ce courriel, pour être tenu informé de son contenu au vu et su des autres destinataires.

2. **Parmi les réponses suivantes, quelles sont celles qui identifient un serveur du domaine egat.fr ?**
   A. http://www.egat.fr
   B. ftp.egat.fr
   C. admin.@egat.fr
   D. www.egat.fr

3. **Que désigne l'extension de fichier .png ?**
   A. Un format d'image
   B. Un format de texte
   C. Un format de page web
   D. Un format de son.

4. **La proposition suivante est-elle vraie ou fausse « un octet permet de coder des entiers positifs au maximum entre 0 et 128 ».**
   A. Vrai B. Faux

5. **En binaire le nombre 67 s'écrit**
   A. `00111110`
   B. `01000011`
   C. `01000101`
   D. `10000000`

6. **La proposition suivante est-elle vraie ou fausse « en notation logique l'expression « VRAI ou FAUX » a pour valeur VRAI ».**
   A. Vrai B. Faux

7. **Le codage des couleurs sur 8 bits permet de coder 256 couleurs**
   A. Vrai B. Faux

Partie 2 : Algorithmique

Pour les questions 8 à 12 nous considérons le problème suivant : On souhaite écrire une fonction en langage C qui retourne une note (entre A et E) en fonction du nombre de bonnes réponses (par tranches de 10 entre 0 et 50). Parmi les programmes ci-dessous, certains réalisent bien cette tâche et sont justes. D'autres comportent des erreurs ou ne réalisent pas cette tâche et sont faux.

Les appels à la fonction se feront par le code suivant :

```
char noteEtudiant ;
noteEtudiant  = note(25) ;
noteEtudiant  = note(52) ;
noteEtudiant  = note(-1) ;
```

8. Le programme suivant réalise-t-il la tâche voulue ?

```
struct bornes {
   int borne_inf, borne_sup;
   char note;
}

char note(int bonnes_reponses) {
   struct bornes bareme[]={
                { 0, 10, 'E'},
                {11, 20, 'D'},
                {21, 30, 'C'},
                {31, 40, 'B'},
                {41, 50, 'A'}};
   int i;
   while (bonnes_reponses != bareme[i]) {
    i++ ;
   }

   return bareme[i].note ;
}
```

A. Oui B. Non

9. Le programme suivant réalise-t-il la tâche voulue ?

```
char note (int bonnes_reponses) {
   if (bonnes_reponses<0 || bonnes_reponses>50)
      return '#';
   else if (bonnes_reponses <= 10)
      return 'E';
   else if (bonnes_reponses <= 20)
      return 'D';
   else if (bonnes_reponses <= 30)
      return 'C';
   else if (bonnes_reponses <= 40)
      return 'B';
   else if (bonnes_reponses <= 50)
      return 'A';
}
```

A. Oui B. Non

10. Le programme suivant réalise-t-il la tâche voulue ?

```
struct bornes {
     int borne_inf, borne_sup;
     char note;
  }

char note(int bonnes_reponses) {
   struct bornes bareme[]={
                { 0, 10, 'E'},
                {11, 20, 'D'},
                {21, 30, 'C'},
                {31, 40, 'B'},
                {41, 50, 'A'}};
   int i;
   for (i=0 ; i < sizeof(bareme)/sizeof(bareme[0]) ; ++i) {
     if (bareme[i].borne_inf >= bonnes_reponses
        && bonnes_reponses >= bareme[i].borne_sup)
      return bareme[i].note;
   }
   return'#';
}
```

A. Oui B. Non

11. Le programme suivant réalise-t-il la tâche voulue ?

```
struct bornes {
   int borne_inf, borne_sup;
   char note;
}

char note(int bonnes_reponses) {
   struct bornes bareme[]={
     { 0, 10, 'E'},
     {11, 20, 'D'},
     {21, 30, 'C'},
     {31, 40, 'B'},
     {41, 50, 'A'}};
   int i;
   for (i=0 ; i < sizeof(bareme)/sizeof(bareme[0]) ; ++i) {
     if (bareme[i].borne_inf <= bonnes_reponses
           && bonnes_reponses <= bareme[i].borne_sup)
     return bareme[i].note;
   }
   return'#';
}
```

A. Oui B. Non

12. Le programme suivant réalise-t-il la tâche voulue ?

```
char note (int bonnes_reponses) {
     if (bonnes_reponses<0 || bonnes_reponses>50)
        return '#';
     if (bonnes_reponses <= 10)
        return 'E';
     if (bonnes_reponses <= 20)
        return 'D';
     if (bonnes_reponses <= 30)
        return 'C';
     if (bonnes_reponses <= 40)
        return 'B';
     if (bonnes_reponses <= 50)
        return 'A';
 }
```

A. Oui B. Non

Partie 3 : Analyse et synthèse de documents

Dans cet exercice, il s'agit d'évaluer votre aptitude à utiliser vos connaissances générales en programmation, pour comprendre un nouveau contexte.
d3.js est une librairie javascript qui permet de manipuler le DOM (Document Object Model). Cette librairie est très efficace dans la manipulation de SVG (Scalable Vector Graphics).

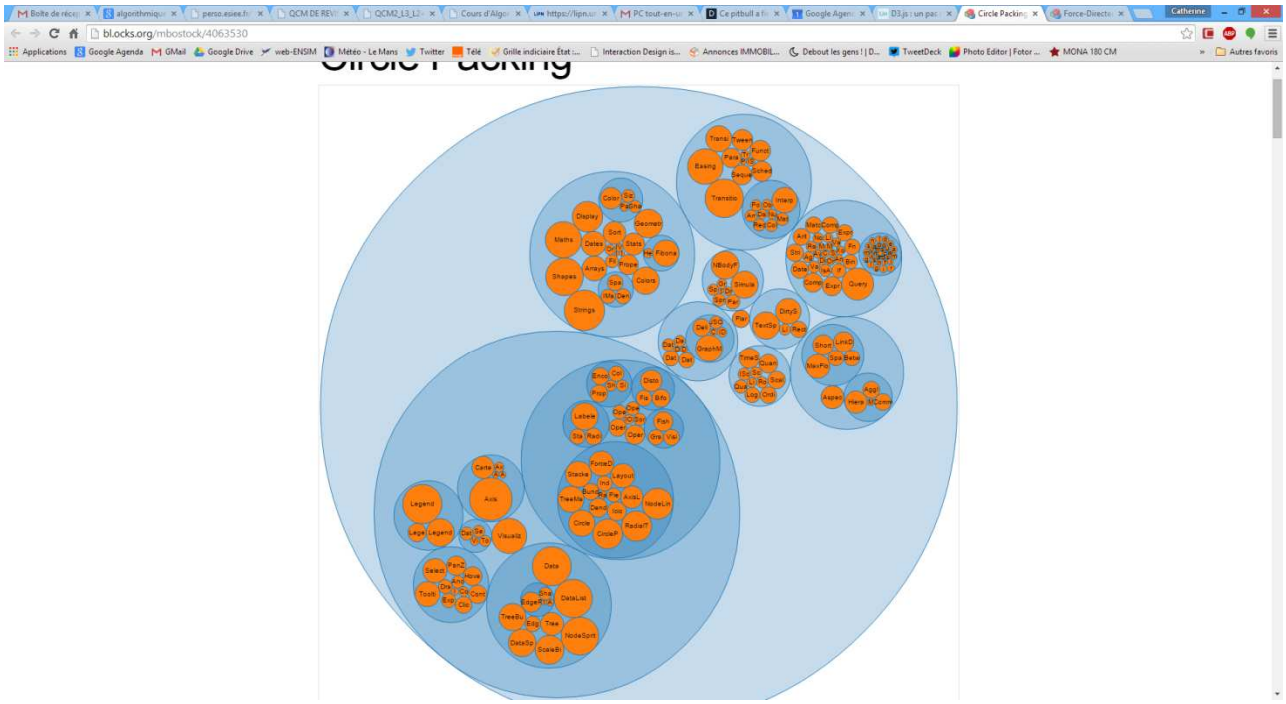

Fig 1 : Exemple de représentation de données par d3.js

En vous **référant à la documentation fournie en annexe**, vous traiterez les questions qui suivent.

13. **La proposition suivante est-elle vraie ou fausse « d3.js est un langage de programmation » ?**

A. Vraie B. Fausse

14. **Est-ce que dans l'exemple 4 de l'annexe, la définition de data aurait pu s'écrire :**

```
var data = [
  { "r":20, "color":"#FDAE6B", "x":190,"y":34,  "name":"cercle 1" },
  { "r":39, "color":"#FDD0A2", "x":150,"y":89,  "name":"cercle 2" },
  { "r":38, "color":"#A1D99B", "x":267,"y":234, "name":"cercle 3" },
  { "r":29, "color":"#31A354", "x":100,"y":34,  "name":"cercle 4" },
  { "r":21, "color":"#3182BD", "x":87, "y":89,  "name":"cercle 5" },
  { "r":25, "color":"#FD8D3C", "x":160,"y":248, "name":"cercle 6" },
  { "r":35, "color":"#A1D99B", "x":78, "y":150, "name":"cercle 7" },
  { "r":32, "color":"#9ECAE1", "x":345,"y":156, "name":"cercle 8" },
  { "r":24, "color":"#C6DBEF", "x":233,"y":117, "name":"cercle 9" }
 ];
```

A. Oui B. Non

15. **Dans l'exemple 4, nous pourrions ajouter les instructions suivantes :**

```
nodes.append("rect")
  .attr("r", function(z) {return z.r})
  .style("fill",function(z){return z.color;})
```

A. Oui B. Non

16. **En synthétisant les éléments donnés dans l'exemple 1 et dans l'exemple 2, le code suivant permet de créer une nouvelle section DIV dans la page HTML comportant du texte**

```
var myDiv = d3.select("body").append("div");

myDiv.style("background-color","#000")
     .attr("id","viz2")
     .text("Hello World 2");
```

A. Vrai B. Faux

17. **DOM signifie Data Object Model ?**

A. Vrai B. Faux

18. **La méthode (ou fonction) .append permet d'ajouter des éléments à des nœuds ?**

A. Vrai B. Faux

19. **SVG est un format de données structuré par balises (nœuds) comme HTML ?**

A. Vrai B. Faux

20. **JSON est un format de données structuré par balises (nœuds) comme HTML ?**

A. Vrai B. Faux

Annexe : Introduction à la librairie javascript : d3.js

Voici un extrait d'un tutoriel en français portant sur la librarie d3.js. issu du site web « undisconnected ». D3.js est une librarie graphique écrite en Javascript crée par Mike Bostock déjà auteur de la librarie Protovis. D3.js est en fait une évolution de protovis, son nom provient de l'abréviation de Data-Driven Document. L'objectif de ce tutoriel est de faire quelques exemples d'utilisation très simple de la librairie.

Exemple 1 : Manipuler ou créer une div avec d3.js

Ce premier exemple montre comment manipuler un nœud DOM (équivalent à une balise HTML). Pour cela il faut deux sortes de code : le code de la page HTML et le code du programme javascript (qui permet de manipuler les balises de la page HTML).

Code HTML :

Une page HTML est structurée en balises <html>, <head>, <body>. Chaque balise a un rôle à jouer. Nous allons ici utiliser la balise qui sert à repérer une division dans la page (donc dans le <body>).

```
<div id='viz'>Hello world</div>
```

Code javascript :

```
// On sélectionne la div avec l'id viz et on lui ajoute une couleur de background noire.
d3.select("#viz").style("background-color", "black");
```

Exemple 2 : Ajouter et manipuler un rectangle SVG

Code javascript :

```
// On crée un élément svg qu'on ajoute au body du html
var svg = d3.select("body").append("svg");

// On ajoute ensuite un rectangle qu'on ajoute dans le svg
svg.append("rect")
   .attr("x",100)
   .attr("y",100)
   .attr("width",200)
   .attr("height",100)
   .style("fill","#000")
   .style("stroke","red")
   .style("stroke-width","10")
```

Fig 2 : résultat de l'ajout d'un rectangle SVG

Exemple 3 : Dessiner plusieurs éléments dans un groupe svg

Code javascript :

```
// On crée un élément svg qu'on ajoute au body du html
var svg = d3.select("body").append("svg");

// On ajoute ensuite un élément de type groupe (g), ainsi qu'à l'intérieur de ce groupe, un cercle et du texte.
var group = svg.append("g")

// On fait un premier cercle dans le groupe
var circle = group.append('circle')
   .attr("cx",150)
   .attr("cy",150)
   .attr("r",45)
   .style("fill","#000")
   .style("stroke","red")
   .style("stroke-width","10")

// On dessine ensuite un rectangle au dessus du cercle
var rect = group.append('rect')
   .attr("x",50)
   .attr("y",50)
   .attr("width",200)
   .attr("height",100)
   .style("fill","#000")
   .style("stroke","red")
   .style("stroke-width","10")

// Puis on dessine 2 cercles dans le rectangle
var circle2 = group.append('circle')
   .attr("cx",200)
   .attr("cy",100)
   .attr("r",20)
   .style("fill","#fff")
   .style("stroke","red")
   .style("stroke-width","10")

var circle3 = group.append('circle')
   .attr("cx",100)
   .attr("cy",100)
   .attr("r",20)
   .style("fill","#fff")
   .style("stroke","red")
   .style("stroke-width","10")
```

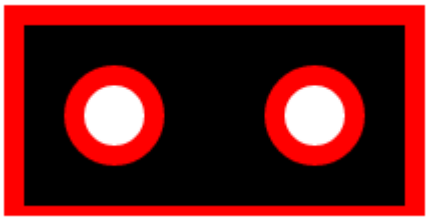

Fig : Résultat du programme de dessin de plusieurs éléments dans un groupe SVG

Exemple 4 : Gérer des données au format JSON et dessiner en fonction de ces données

Dans ce dernier exemple, nous allons construire des nodes qui sont en fait des groupes svg ayant la class “node”. Ensuite, on va dessiner, suivant les données (variable data) un cercle et afficher le nom de ce cercle dans celui -ci.

Code javascript :

```
var data = [
    {"x":190,"y":34, "r":20,"color":"#FDAE6B","name":"cercle 1"},
    {"x":150,"y":89, "r":39,"color":"#FDD0A2","name":"cercle 2"},
    {"x":267,"y":234,"r":38,"color":"#A1D99B","name":"cercle 3"},
    {"x":100,"y":34, "r":29,"color":"#31A354","name":"cercle 4"},
    {"x":87, "y":89, "r":21,"color":"#3182BD","name":"cercle 5"},
    {"x":160,"y":248,"r":25,"color":"#FD8D3C","name":"cercle 6"},
    {"x":78, "y":150,"r":35,"color":"#A1D99B","name":"cercle 7"},
    {"x":345,"y":156,"r":32,"color":"#9ECAE1","name":"cercle 8"},
    {"x":233,"y":117,"r":24,"color":"#C6DBEF","name":"cercle 9"}
];

var svg = d3.select("body").append("svg");

var nodes = svg.selectAll(".node")
   .data(data)
   .enter()
     .append("g") // le nœud g désigne un groupement de formes SVG
     .attr("class","node")
     .attr("transform", function(d) { return "translate(" + d.x + ","
+ d.y + ")"; });


   nodes.append("circle")
     .attr("r", function(d) {return d.r})
     .style("fill", function(d){return d.color;})

   nodes.append("text")
      .attr("dy", ".3em")
      .style("text-anchor", "middle")
      .text( function(d) { return d.name; });
```

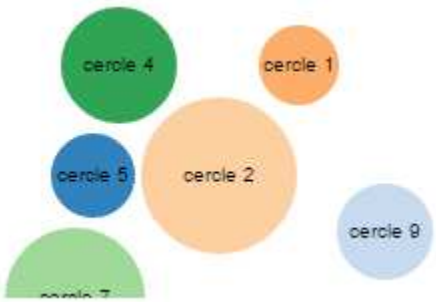

Fig 4 : Résultat du programme de gestion de données au format JSON

Le dernier exemple introduit plusieurs nouveaux concepts : selectAll(), data(), et enter().
selectAll() : La méthode de sélection selectAll permet, comme son nom l'indique de sélectionner les éléments définis par le sélecteur passé en argument. selectAll("p") va sélectionner tous les éléments <p> présents dans le DOM.
data() : La méthode data() permet de "binder" (attacher) des données à des éléments sélectionnés par la méthode selectAll(). Ces données une fois attachées seront utilisables dans la suite du chainage d'appels à des fonctions et envoyées en paramètre des fonctions crées localement (dans le corps des instructions) comme ici dans la fonction « function() » :

```
nodes.append("circle")
    .attr("r", function(d) {return d.r})
     .style("fill",function(d){return d.color;})
```

enter() : Cette fonction va " binder " les données à des éléments et créer les éléments sélectionnés par selectAll(). Si vous n'utilisez pas enter() alors aucun élément ne sera créé. Seuls les éléments existants seront utilisés et donc s'ils sont inexistants, rien ne s'affichera.