|  |  |  |
| --- | --- | --- |
| **OFPPT** | **مــــــكــــــتــــــب الــــــتــــــكــــــويــــــن الــــــمــــــهــــــنــــــي و إنــــــعــــــاش الــــــشــــــغــــــل** | |
| **Office de la Formation Professionnelle et de la Promotion du Travail** | |
| **Direction Recherche et Ingénierie de la Formation** | | |
| **Examen de passage à la 2ème année**  **Session Juillet 2018** | | |
| **Filière  :** **Techniques de Développement Informatique** | | **Epreuve  : Synthèse** |
| **Niveau  : TS** | | **Variante  : V1** |
| **Durée  : 5 heures** | | **Barème  : / 120 pts** |

|  |
| --- |
| **Eléments de réponse** |

|  |  |
| --- | --- |
| **Partie I : Théorie** | **40 pts** |
|  | |
| **Dossier 1 : L’essentiel en technologies de l’information** | **12 pts** |
|  | |
| 1. **0.5 pt pour chaque conversion** | **6 pts** |
| |  |  |  |  | | --- | --- | --- | --- | | **Binaire** | **Octal** | **Décimal** | **Hexadécimal** | | 11000011 | **303** | **195** | **C3** | | **110111** | 67 | **55** | **37** | | **1000011** | **103** | 67 | **43** | | **1100111** | **147** | **103** | 67 | | |
|  | |
| 1. La fonction f : | **1.5 pts** |
| Simplification analytique : | **1.5 pts** |
| 1. Simplification moyennant le tableau de Karnaugh :  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  | **1** | 0 | **1** | 0 | |  | **1** | 0 | **1** | 0 | | **3 pts** |

|  |  |  |
| --- | --- | --- |
| **Dossier 2 : Analyse et conception orientée objet** | **16 pts** | |
|  | | |
| 1. Diagramme de cas d’utilisation : **0.25 pt pour chaque élément (acteur, Cas d’utilisation et association)** | **7.5 pts** | |
|  | | |
| 1. Diagramme de classes : **0.5 pt pour chaque élément (classe et association)** | **8.5 pts** | |
|  | | |
| **Dossier 3 : Programmation structurée** | **12 pts** | |
|  | | |
| Algorithme Gestion des villes |  | |
| Structure Ville  Nom : Chaine  Population : Entier  Pays : Chaine  FinStructure | **1 pt** | |
| Constante nMax 🡨 20 |  | |
| Variables villes[nMax] : Villes | **1 pt** | |
| n, i  populationTotal,  populationMax,  indiceVillePeuplee : Entier  nomPays : Chaine  Début |  | |
| Répéter  Ecrire("Nombre de villes ? ")  Lire(n)  TantQue(NON(n > 0 ET n <= nMax)) | **1 pt** | |
| Pour i 🡨 0 jusqu’à n - 1  Ecrire("Ville ", i + 1, " : \n")  Ecrire("Nom : ")  Lire(villes[i].Nom)  Ecrire("Population : ")  Lire(villes[i].Population)  Ecrire("Pays : ")  Lire(villes[i].Pays)  FinPour | **2 pts** | |
| Ecrire("Pays dont vous voulez calculer la population totale : ")  Lire(nomPays)  populationTotale 🡨 0  Pour i 🡨 0 jusqu’à n - 1  Si(nomPays = villes[i].Pays)  populationTotale += villes[i].Population  FinSi  FinPour  Ecrire("La population totale est : ", populationTotale) | **3 pts** | |
| populationMax 🡨 0  Pour i 🡨 0 jusqu’à n - 1  Si(populationMax < villes[i].Population)  populationMax 🡨 villes[i].Population  indiceVillePeuplee 🡨 i  FinSi  FinPour  Ecrire("Ville la plus peuplée : ", villes[indiceVillePeuplee].Nom) | **4 pts** | |
| Fin |  | |
|  | | |
| **Partie II : Pratique** | | **80 pts** |
|  | | |
| **Dossier 1 : Programmation structurée** | | **16 pts** |
|  | | |
| 1. f(275) 🡪 3 | | **2 pts** |
| 1. f(27) 🡪 2 | | **2 pts** |
| 1. f(2) 🡪 1 | | **2 pts** |
| |  | | --- | |  | | | |
| |  | | --- | |  | | | |
| **Dossier 2 : Programmation événementielle et orientée objet** | | **64 pts** |
|  | | |
| class Pays  {  //1. 1.5 pt  private string \_Nom;  public string Nom  {  get { return \_Nom; }  set  {  //1.a 3 pts  if (value.Length < 4)  throw new Exception("Le nom doit contenir au moins 4 caractères");  foreach (char caractère in value)  {  if (char.IsDigit(caractère))  throw new Exception("Le nom ne peut pas contenir des chiffres");  }  \_Nom = value;  }  }  //1.(suite) 1.5 pt  private List<Ville> \_Villes = new List<Ville>();  public List<Ville> Villes  {  get { return \_Villes; }  set { \_Villes = value; }  }  //1.(suite) 1.5 pt  private Ville \_Capitale;  public Ville Capitale  {  get { return \_Capitale; }  set  {  //1.b 3 pts  Ajouter(value);  //La méthode vérifie l'existence de la ville  \_Capitale = value;  }  }  //2.a 1.5 pt  public Pays(string nom)  {  Nom = nom;  }  //2.b 2 pts  public Pays(string nom, Ville capitale)  {  Nom = nom;  Capitale = capitale;  }  //3 2 pts  public int NombreVilles  {  get { return Villes.Count; }  }  //4 3 pts  public int Rechercher(String nomVille)  {  foreach (Ville ville in Villes)  if (ville.Nom == nomVille) return Villes.IndexOf(ville);  return -1;  }  //5 3 pts  public void Ajouter(Ville ville)  {  if (Rechercher(ville.Nom) == -1) Villes.Add(ville);  }  //6 4 pts  public void Supprimer(String nomVille)  {  int indiceVille = Rechercher(nomVille);  if (indiceVille != -1)  {  Villes.RemoveAt(indiceVille);  if (Capitale.Nom == nomVille) Capitale = null;  }  }  //7 5 pts  public List<Ville> TrierParPopulation(string ordre)  {  List<Ville> villes = new List<Ville>();  villes.AddRange(Villes);  if (ordre.ToLower() == "croissant")  villes.Sort(new PopulationCroissantComparer());  else if (ordre.ToLower() == "décroissant")  villes.Sort(new PopulationDécroissantComparer());  return villes;  }  //8 3 pts  public int PopulationTotale()  {  int populationTotale = 0;  foreach (Ville ville in Villes)  populationTotale += ville.Population;  return populationTotale;  }  }  //7(suite) 5 pts  class PopulationCroissantComparer : IComparer<Ville>  {  public int Compare(Ville x, Ville y)  {  return x.Population.CompareTo(y.Population);  }  }  //7(suite) 5 pts  class PopulationDécroissantComparer : IComparer<Ville>  {  public int Compare(Ville x, Ville y)  {  return -x.Population.CompareTo(y.Population);  }  } | | |
|  | | |
|  | | |
| public partial class Gestion\_des\_pays : Form  {  //9.a 1 pt  List<Pays> LesPays = new List<Pays>();  public Gestion\_des\_pays()  {  InitializeComponent();  }  //9.b 1 pt  private void Gestion\_des\_pays\_Load(object sender, EventArgs e)  {  //9.b.i 6 pts  Pays maroc = new Pays("Maroc");  maroc.Ajouter(new Ville("Casablanca", 3359000));  maroc.Ajouter(new Ville("Rabat", 577000));  maroc.Ajouter(new Ville("Fès", 1112000));  maroc.Ajouter(new Ville("Tanger", 974000));  maroc.Ajouter(new Ville("Marrakech", 928850));  maroc.Capitale = maroc.Villes[1];  LesPays.Add(maroc);  Pays tunisie = new Pays("Tunisie");  LesPays.Add(tunisie);  Pays jordanie = new Pays("Jordanie");  jordanie.Ajouter(new Ville("Zarqa", 395227));  jordanie.Ajouter(new Ville("Irbid", 250645));  jordanie.Capitale = new Ville("Amman", 994199);  LesPays.Add(jordanie);  //9.b.ii 3 pts  comboBox\_Pays.DataSource = LesPays;  comboBox\_Pays.DisplayMember = "Nom";  }  //9.c 4 pts  private void comboBox\_Pays\_SelectedIndexChanged(object sender, EventArgs e)  {  Pays pays = comboBox\_Pays.SelectedItem as Pays;  listBox\_Villes.Items.Clear();  if (pays.NombreVilles != 0)  {  foreach (Ville ville in pays.Villes)  {  //9.c.i 2 pts  if (pays.Capitale != null && ville.Nom == pays.Capitale.Nom)  listBox\_Villes.Items.Add("--> " + ville);  else  listBox\_Villes.Items.Add(ville);  }  }  else  //9.c.ii 2 pts  listBox\_Villes.Items.Add("Aucune ville !!!");  }  //9.d 4 pts  private void button\_Trier\_Click(object sender, EventArgs e)  {  Pays pays = comboBox\_Pays.SelectedItem as Pays;  if (pays.NombreVilles != 0)  {  string ordre = "croissant";  List<Ville> villes;  if (radioButton\_Décroissant.Checked) ordre = "décroissant";  villes = pays.TrierParPopulation(ordre);  listBox\_Villes.Items.Clear();  foreach (Ville ville in villes)  {  if (pays.Capitale != null && ville.Nom == pays.Capitale.Nom)  listBox\_Villes.Items.Add("--> " + ville);  else  listBox\_Villes.Items.Add(ville);  }  }  }  //9.e 4 pts  private void button\_OrdreInitial\_Click(object sender, EventArgs e)  {  comboBox\_Pays\_SelectedIndexChanged(comboBox\_Pays, new EventArgs());  }  //9.f 3 pts  void Gestion\_des\_pays\_FormClosing(object sender, FormClosingEventArgs e)  {  if (MessageBox.Show("Voulez-vous vraiment fermer l'application ?",  "Attention",  MessageBoxButtons.YesNo,  MessageBoxIcon.Question) == DialogResult.No)  e.Cancel = true;  }  } | | |