**1ER PARCIAL**

|  |  |  |
| --- | --- | --- |
| **Tema** | **Forma de evaluación** | **Puntaje** |
| **TODOS**  Variables simples  Arreglos  Registros  Archivos de texto y binarios | **Seguimiento de Algoritmo**  **Tiempo previsto: 15 min – 2pts**   1. Escriba en el cuadro de abajo los valores de salida del siguiente algoritmo y 2. ADJUNTE UNA IMAGEN CON LAS TABLAS DE VALORES DE LAS VARIABLES SEGUIDAS   **Atención**, los valores de ingreso cambian según la **PRIMERA LETRA de su APELLIDO**  Tome los que correspondan a su letra real de comienzo.  **"A" a "F" incl./ "G" a "L" incl./ "M" a "R" incl. /  "S" a "Z" incl.**  **Variables       Variables       Variables        Variables**  **A     B                  A     B                  A     B                 A     B**         3     1                   1     3                     1     1                    3     3         2     2                   2     2                    2     2                   2     2         1     3                   3     1                     3     3                   1     1  por ej. si usted se apellida **Pérez**, trabajará con los valores **a y b**de la**3ra columna**    Suponga un vector **Vec de 4 posic** que arranca en **0 (CERO)** en cada una  (no haga caso a la sintaxis, está escrito como pseudocódigo)    Se sugiere armar una tabla con columnas para las siguientes variables (tantas filas como necesite)            A         B          i          j        Vec[0]       Vec[1]        Vec[2]       Vec[3]    **Algoritmo Pseudocódigo**  **PARA**- **FIN PARA**= For ,  **INGRESAR**= cin,  **ESCRIBIR**= cout,  **FINLÍNEA**= endl      PARA (i=0, i<=2, i++)   |      INGRESAR A;   |      INGRESAR B;   |     Vec[i] =Vec[i] + A;   |      PARA (j=2, j>i, j--)   |      |     Vec[j] =Vec[j] + B;   |      FIN PARA   FIN PARA   PARA (i=0, i<=3, i++)   |      ESCRIBIR Vec[i], FINLÍNEA   FIN PARA **Seguimiento / Salidas obtenidas**  “A” a “F”   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **A** | **B** | **i** | **j** | **Vec[0]** | **Vec[1]** | **Vec[2]** | **Vec[3]** | | 3 | 1 | 0 | 2 | 0 | 0 | 0 | **0** | |  |  |  | 1 | **3** | 1 | 1 |  | | 2 | 2 | 1 | 2 |  | **3** | 3 |  | | 1 | 3 | 2 |  |  |  | **4** |  |   “G” a “L”   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **A** | **B** | **i** | **j** | **Vec[0]** | **Vec[1]** | **Vec[2]** | **Vec[3]** | | 1 | 3 | 0 | 2 | 0 | 0 | 0 | **0** | |  |  |  | 1 | **1** | 3 | 3 |  | | 2 | 2 | 1 | 2 |  | **5** | 5 |  | | 3 | 1 | 2 |  |  |  | **8** |  |   “M” a “R”   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **A** | **B** | **i** | **j** | **Vec[0]** | **Vec[1]** | **Vec[2]** | **Vec[3]** | | 1 | 1 | 0 | 2 | 0 | 0 | 0 | **0** | |  |  |  | 1 | **1** | 1 | 1 |  | | 2 | 2 | 1 | 2 |  | **3** | 3 |  | | 3 | 3 | 2 |  |  |  | **6** |  |   “S” a “Z”   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **A** | **B** | **i** | **j** | **Vec[0]** | **Vec[1]** | **Vec[2]** | **Vec[3]** | | 3 | 3 | 0 | 2 | 0 | 0 | 0 | **0** | |  |  |  | 1 | **3** | 3 | 3 |  | | 2 | 2 | 1 | 2 |  | **5** | 5 |  | | 1 | 1 | 2 |  |  |  | **6** |  | | 2 pts |
|  |  |  |
| **Conceptos básicos:**  Algoritmos, variable, etc.  **Comandos:**  Asignación externa de Entrada  Asignación externa de Salida  Asignación interna  Alternativa  Alternativa múltiple  Ciclo exacto  Ciclo precondicional  Ciclo postcondicional  **Funciones:**  void  no void  **Estructuras de datos:**  Variables simples  Arreglos  Registros  Archivos de texto y binarios | **Pregunta tipo múltiple choice**  **Tiempo previsto: 10 min – 1pt**  **Invocación Función no void**  Se desea invocar a una función **Buscar no void de tipo booleana** que como parámetros recibe **un vector, un tope, un Cod** de 3 dígitos a buscar,  la misma **devuelve por parámetro de referencia la pos** donde encuentra el Cod si está,  y **retorna True o False** según si encuentra o no el Cód.  Cuál de las siguientes invocaciones es correcta?  pos= Buscar (Vec, tope, Cod, encontro);  if (Buscar (Vec, tope, Cod, pos, encontro)!=-1 {  if Buscar (Vec, tope, 140, pos, encontro) {  if Buscar (Vec, tope, 140, pos) {  if (Buscar (Vec, tope, 140)!=-1) {  enc= Buscar (Vec, tope, Cod, encontro);  Ninguna de las expuestas  **1pt**  **Respuestas**  **if (Buscar (Vec, tope, 140, pos)) {**  o  **Ninguna de las expuestas** (puesto que en las opciones involuntariamente no quedaron los paréntesis de la condición if, esta se toma correctamente como válida. La de arriba también ya que se aludía a la condición lógica de los parámetros no a la sintaxis como ser falta de ; etc.) | 1 pt |
| **TODOS**  Variables simples  Arreglos  Registros  Archivos de texto y binarios | **Desarrollo de algoritmo modular**  **Tiempo previsto: 45 min - 5 pts**  **Desarrolle el algoritmo solicitado**  Una casa de venta de electrodomésticos desea realizar un control de stock de determinadas cocinas y ordenar reposición de las que tengan su stock actual por debajo del mínimo requerido.  a) Desarrolle un módulo void **ControlarRepoCocinaTipoY** que recibe **entre otros** los siguientes parámetros:  -un vector **VP** de tipo tProd ya cargado con los no más de 200 productos que comercializa, que cuenta con la siguiente estructura  **Tipo** (Tipo de producto, 1 a 25),  **CodP** (Código de Producto, 4 dígitos),  **Stk** (Stock actual del producto),  **Stkmin** (Stock mínimo que sería recomendable que tenga dicho producto);  -un **Tipo de producto** (entre 1 y 25, el cual corresponde al tipo de cocinas) que se quiere filtrar.  ***Debe***  *-****generar un archivo “RepoCocina.dat”*** *con los datos del Tipo de producto recibido por parámetro (que son las cocinas que se quiere filtrar) y que necesitan reposición de stock (es decir su Stk se encuentra por debajo del Stkmin), conteniendo la siguiente estructura (del tipo de datos tRepo):* ***CodP*** *y* ***Cant a reponer*** *(que será la necesaria para llevar el stock actual al Stkmin);*  *-****devolver por referencia el Código del Producto*** *con menor Stock actual (pero mayor a cero)* ***y su posición dentro del vector*** *(sin filtrar por tipo de producto recibido; hay un solo menor);*  *Agréguele al módulo el/los parámetro/s adicional/es que requiera.*  b) desarrolle los **struct** de los tipos de datos  ***tProd*** *necesario para el del vector* ***VP****, y*  ***tRepo*** *necesario para el archivo de* ***Reposición****.*  **Solución posible**  **a)**  **Void ControlarRepoCocinaTipoY** (tProd VP[], int tope, int Tipo, int & minCodP, int & minpos) {  FILE\* aRepo = **fopen** (“RepoCocina.dat”, “wb”);  tRepo RR;  int i;  bool pvez=true;  for (i=0;i<=tope;i++) {  if ((VP[i].Tipo==Tipo) && (VP[i].Stk < VP[i].Stkmin)) {  RR.Codp = VP[i].Codp;  RR.CantARep = VP[i].Stkmin- VP[i].Stk;  **fwrite** (&RR, **sizeof** (RR), 1, aRepo);  }  **if** (VP[i].Stk>0) {  **if** (pvez) o (VP[i].Stk < minStk) {  minStk = VP[i].Stk;  minCodP = VP[i].CodP;  minpos = i;  pvez = false;  }  }  **fclose** (aRepo);  };  **b)**  **struct** tProd{  **int** Tipo;  **int** CodP;  **int** Stk;  **int** Stkmin;  };  **struct** tRepo{  **int** CodP;  **int** CantARep;  }; | 5 pts  (4 pts + 1pt) |
| **TODOS**  Variables simples  Arreglos  Registros  Archivos de texto y binarios | **Desarrollo de algoritmo modular para posibilidad de promoción**  **Tiempo previsto: 20 min - 2 pts**  **Desarrolle el algoritmo solicitado**  a) Desarrollar un módulo no void **CorrerDatos** que recibe **entre otros** los siguientes parámetros:  -un vector **VLeg** de 150 posiciones que contiene únicamente **Legajos** de tipo entero;  -un valor **tope** < 150 con la posición del último **Legajo** cargado en el vector;  -un valor **pos** <= tope;  -un nuevo **Legajo** a “insertar” en la posición **pos** del vector.  ***Debe***  *-Insertar en la posición pos de VLeg, el* Legajo recibido sin perderse los datos ya cargados en el vector. C*omo* ***pos*** *es <=* ***tope****, en dicha posición y las que siguen hasta tope habrá Legajos ya cargados; por lo tanto para poder insertar deberá realizar el corrimiento de los elementos necesarios (copiando cada uno debajo de su posición actual).*  *-Retornar* ***true*** *en caso en que el vector se haya llenado; caso contrario retornar* ***false****.*    *Agréguele al módulo el/los parámetro/s adicional/es que requiera.*  bool **CorrerDatos** (int VLeg,int & tope,int pos,int Legajo) {  for (i=tope;i>=pos;i--) {  VLeg[i+1]= VLeg[i];  }  VLeg[pos]=Legajo;  tope++;  **if** (tope=149) {  Return true  }  else {  Return false  };  }; | 2 pts  (para  posibilidad de promoción) |
|  |  |  |