Nombre completo: Enrique Mena Camilo
Fecha: jueves 11 de marzo de 2021
Email: enriquemece97@gmail.com Teléfono (10 dígitos): 733 118 7123

PRUEBA DE CONOCIMIENTOS TÉCNICOS.

Responde a los siguientes reactivos como parte del proceso de reclutamiento para la vacante **Desarrollador de sistemas embebidos**.

De ser posible, envía un avance antes del **jueves a las 23.00 hrs.**
Cualquier avance entregado será considerado en la evaluación.

1. **Diseña, codifica y comenta el código de una función en C++ que sea capaz de asignar dinámicamente la memoria a un conjunto de datos de tipo array que almacene nombres de usuario, reservando y liberando la memoria para un conjunto de 10 registros.**

Coloca una captura de pantalla de tu código desde tu compilador.

```cpp
#include <iostream>
#include <string.h>

using namespace std;

void registroNuevo(); //Prototipo de función para registro de usuario nuevo

int main()
{

    //Solicitud de registro de usuario nuevo 10 veces
    for(int i=0; i<10; i++)
    {
        registroNuevo();
    }
    return 0;
}

//Función para generar el registro de usuario nuevo
void registroNuevo()
{
    char temp[30]; //Variable temporal para almacenar nombre de usuario
    char *usuario; //Apuntador para reservar memoria para nombre de usuario
    int sizeUsuario; //Variable para almacenar longitud del nombre de usuario

    cout<<"Ingrese nombre de usuario: ";
    cin.getline(temp,30,'\n'); //Lectura de nombre de usuario
    sizeUsuario = strlen(temp); //Determinación de longitud de nombre de usuario

    usuario = new char[sizeUsuario]; //Asignamos memoria específica para usuario
    strcpy(usuario, temp); //Copia contenido de temporal a usuario (dinámico)
    cout<<usuario<<" tamaño: "<<sizeUsuario<<endl; //Uso del arreglo dinámico
    delete [] usuario; //Libera memoria
}
```

Adjunta tu archivo .cpp en el mail de respuesta de la prueba.

2. **Clasifica o describe el tipo de error de los siguientes ejemplos en el manejo de memoria:**

```
1. int n;
2. int *ptr = &n;
3. ptr = 9;
```

1) Describe el error: ptr es una variable de tipo puntero, eso implica que dicha variable almacena la dirección de memoria de la variable a la que apunta, en este caso la dirección de memoria de n*, por ello, la asignación de la línea 3 marca un error, ya que se está intentando almacenar un valor del tipo entero en una variable que almacena una dirección de memoria. Si lo que se desea es cambiar la dirección de memoria a la que apunta dicho puntero, se tendría que hacer algo similar a* ptr = &otraVariable*, pero si lo que se desea es cambiar el valor de la variable a la que apunta el puntero se deberá escribir algo similar a* *ptr = 9.

```
1. char *a;
```

```
2. a = new char[20];
3. …
4. delete a;
5. …
6. if(a != NULL)
7.          strcpy(a, "Una cadena");
```

2) Describe el error: Cuando se trabaja con arreglos dinámicos, el uso del comando delete *debe estar acompañado de un par de corchetes vacíos, esto para asegurarse que se libera todo el espacio de memoria reservado para el arreglo. Por esto, la línea 4 debería ser de esta forma* delete [] a.

```
1. int a = 10;
2. int *ptri = NULL;
3. double x = 5.0;
4. double *ptrf = NULL;
5. ...
6. ptri = &a;
7. ptrf = &x;
8. ptrf = ptri;
```

3) Describe el error: En la línea 8 se está intentando asignar un puntero de tipo entero a un puntero de tipo double, lo cual no es posible. Se tiene que asignar una dirección de memoria del tipo double o bien otro puntero del mismo tipo.

```
1. char lista[4] = {'a', 'b', 'c', 'd'};
2. char *ptrLista = lista;
3. char *ptrLista = NULL;
```

4) Describe el error: el puntero de nombre ptrLista *se está definiendo dos veces. En la línea 2 se define el puntero de tipo char que apunta al array* lista*, y en la línea 3 se intenta definir nuevamente un puntero con el mismo nombre. Si se desea cambiar el valor del puntero se debe hacer sin agregar el comando* char.

3. **El siguiente ejemplo está diseñado utilizando QML versión 2.0. Completa los siguientes puntos:**
    a) Enuncia todos los elementos gráficos que identifiques que lo componen (botones, tablas, charts, etc).
        i) Botones
        ii) Checkbox
        iii) Texto
        iv) Gráficos (charts)
        v) Series
        vi) Leyenda de gráfica

vii) Rectángulos

b) Describe los métodos (en JS) del protocolo XMLHttpRequest que permiten obtener los datos desde una API externa del índice NASDAQ-100.

i)

4. **Revisa el siguiente ejemplo: [bluetooth-heartrate-game-example] y explica qué cambios realizarías para cubrir los siguientes requerimientos:**

1) Expandir el número de conexiones a un panel con 5 lecturas (dispositivos) simultáneas.

   a) Crear múltiples instancias del objeto QLowEnergyController, cada una conectada a un dispositivo diferente.

2) Cambiar el color de la interfaz conforme se vaya aumentando en 10 el valor de la frecuencia leída.

   a) Retirar *readOnly* de la propiedad *backgroundColor* configurada en el archivo *GameSettings.qml*

   b) En el archivo *Measure.qml*, dentro de la función *start()* vincularía la propiedad *GameSettings.backgroundColor* a una función del archivo *devicehandler.cpp*, la cual contendría la lógica para generar un número, asociado al color de la interfaz, cada vez que se aumenten la frecuencia cardiaca.

3) *(Adicional)* En caso de tener acceso a dispositivos bluetooth (no necesariamente de frecuencia cardíaca), nos gustaría que instales Qt desde el siguiente enlace [download-qt-installer], accedas al ejemplo en *"Welcome mode" > "Examples"* e implementa tu propuesta del punto 1.

*El punto 3) adicional puede entregarse después para ser comentado en las próximas entrevistas técnicas.

5. **Manejo de threads:**

    a. Tomando un ejemplo de los que vienen con QML por default, por ejemplo un reloj de manecillas, el código del backend del reloj pásalo a un thread y haz que no corra en el hilo principal.
    b. Instancia múltiples threads del creado anteriormente e imprime en terminal la hora de cada reloj en el thread de forma asíncrona y que diga: Yo thread X tengo Y hora.
    c. Utilizar gridLayout de QML para crear una matriz de relojes donde cada reloj es alimentado por la hora de un thread y cada thread sabe identificar a su reloj.