1) Vamos implementar en C++ con header (.h) y .cpp una lista enlazada doblemente enlazada de enteros con centinelas que llamaremos Lista_dob_enl_cen.

Crear una carpeta vacía llamada ListaDoble.

- Ejecutar VSCODE y abrir esa carpeta (open folder)
- Crear los archivos (Lista_dob_enl_cen.h, Lista_dob_enl_cen.cpp)
- Crear un archivo llamado prueba.cpp donde harán la prueba del tipo Lista_dob_enl_cen,
- cambiar el tasks.json : "${file}", por "${workspaceFolder}*.cpp", si quiere ejecutar el programa con VScode. Si no utilice el comando en terminal:
  - g++ -o prueba prueba.cpp Lista_dob_enl_cen.cpp

No vean ningún material (deben pensar ustedes)

Creen el archivo **Lista_dob_enl_cen.h** que contendrá la estructura de datos y los protocolos de las operaciones:

crear una lista vacía, insertar un elemento en una posición (puede ser de úlmino si la posición es igual al número de elementos, eliminar un elemento en una posición, imprimir la lista, verificar si es vacía.

Creen el archivo Lista_dob_enl_cen.cpp

En el archivo de prueba, ir probando las funciones

Lista_dob_enl_cen.h:

```
struct Nodo {
  int data;
  Nodo *anterior;
  Nodo *proximo;
};
struct Lista_dob_enl{
  Nodo* cabeza = nullptr;
  Nodo* cola = nullptr;
  int num_elem=0;
} ;
```

```
Lista_dob_enl crear_lista_vacia();
void insertar(Lista_dob_enl &, int , int pos);
void imprimir(const Lista_dob_enl &);
void eliminar(Lista_dob_enl &, int );
bool esVacia(const Lista_dob_enl &);
```

Archivo **Lista_dob_enl_cen.cpp:**

Hay que incluir al comienzo:

#include <iostrem>

#include “Lista_dob_enl_cen.h”

```
Lista_dob_enl_cen crear_lista_vacia(){
  Lista_dob_enl_cen p;
  p.cabeza = new Nodo{0,nullptr, nullptr};
  p.cola = new Nodo{0,p.cabeza, nullptr};
  (p.cabeza)->proximo = p.cola;
  p.num_elem =0;
  return p;
}
```

```
bool esVacia(const Lista_dob_enl &lista){
  return (lista.num_elem == 0);
}
```

```
void insertar(Lista_dob_enl &lista, int x, int pos){
  // Inserta x en la posición pos de lista
  if (pos <0 || pos > lista.num_elem) return;
  Nodo* cursor = Obtener_Ref(lista,pos);
  Nodo* nuevo = new Nodo{x,cursor->anterior,cursor};
  (cursor->anterior)->proximo = nuevo;
  cursor->anterior = nuevo;
  (lista.num_elem)++;
}
```

```
Nodo* Obtener_Ref(Lista_dob_enl &lista, int pos){
  // pos debe estar entre 0 y numero de elementos
// de la lista
  Nodo* cursor = (lista.cabeza)->proximo;
  int posicion=0;
  while (posicion != pos){
    cursor = cursor->proximo;
    posicion++;
  }
  return cursor;
}
```

Funcion eliminar:

```
void eliminar(Lista_dob_enl &lista, int pos){
  if (pos <0 || pos >= lista.num_elem) return;
  Nodo* cursor = Obtener_Ref(lista,pos);
  (cursor->anterior)->proximo = cursor->proximo;
  (cursor->proximo)->anterior  = cursor->anterior;
  (lista.num_elem)--;
}
```

```
void imprimir(const Lista_dob_enl &lista){
  // imprime lista
  Nodo* p = lista.cabeza->proximo;
  while (p != lista.cola){
     std::cout << p->data << " ";
     p = p->proximo;
  }
  std::cout << std::endl;
}
```

Ahora en el archivo prueba.cpp:

```
int main(){

    Lista_dob_enl_cen lista = crear_lista_vacia();

    insertar(lista, 3000, 0);
    insertar(lista, 1, 1);
    insertar(lista, 2, 1);
    eliminar(lista,1);
    eliminar(lista,1);
    imprimir(lista);
    return 0;

}
```