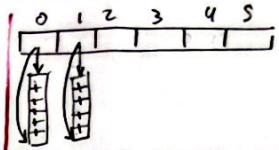
IKI. K.



Vector de colas:
```
struct VecCola{
    NodoCola* frente;
    NodoCola* fin;
};
```

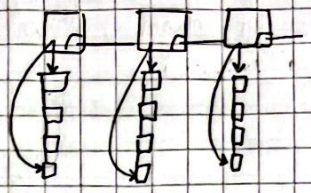
```
struct NodoCola{
    int info;
    NodoCola* sgte;
}
```

Vector de colas: `VecCola vectorColas[6];`

Final AyED 31/07/23

Lista de colas

```
struct NodoCola{
    char nombre[41];
    NodoCola* sgte;
};
```



```
struct InfoLista{ // "TipoInfoPrioridades", me parece más entendible así
    NodoCola* frente;
    NodoCola* fin;
};

struct NodoLista{ // "NodoPrioridad"
    InfoLista info;
    NodoLista* sgte;
};
```

1.
```
NodoLista* crearListaDeColas(){ // "crearColaPrioridad"
    NodoLista* listaPrioridades = NULL; // inicializo un primer Nodo
    InfoLista info;

    for(int i=0; i<5; i++){ // creo los nodos de la lista
        info.frente = NULL; // inicializo una cola
        info.fin = NULL;
        agregar(listaPrioridades, info);
    }
}
```

Puntero al primer nodo de la lista

```
void agregar(NodoLista* &lista, InfoLista info){
    NodoLista* nuevo = new Nodo();
    nuevo -> info = info;
    nuevo -> sgte = NULL;
    if(lista == NULL){ // si está vacía, el nuevo será el primer nodo
        lista = nuevo;
    }else{             // si no está vacía
        Nodo* aux = lista;
        while(aux->sgte != NULL){ // recorro hasta el final (sabemos que el último apunta a la nada)
            aux = aux -> sgte;
        }
        aux->sgte = nuevo;
    }
}
```

"InsertarEnPrioridad"

2.
```
void agregarPaciente(NodoLista* &listaPrioridades, int Prioridad, char paciente[]){
    NodoLista* aux = listaPrioridades;
    for(int i=0; i<prioridad, i++){ // voy hasta el nodo de esa prioridad
        aux = aux -> sgte;
    }
    queue(aux->info.frente, aux->info.fin, paciente);
}
```

3.
```
NodoLista* elementoMinimo(NodoLista* listaPrioridades){
    NodoLista* aux = listaPrioridades;
    while(aux != NULL && aux->info.frente == NULL){
        aux = aux -> sgte;
    }
    return aux;
}
```

4.
```
char* quitarMinimo (NodoLista* &listaPrioridades) {
    NodoLista* aux = elementoMinimo (listaPrioridades);
    Char paciente[41];
    paciente = unqueue(aux->info.frente, aux->info.fin);
    return paciente;
}
```

5.
```
int prioridadVacia (NodoLista* listaPrioridades){
    NodoLista* aux = listaPrioridades;
    while(aux!=NULL && aux->info.frente==NULL){
        aux = aux->sgte;
    }
    IF(aux==NULL){
        return 1;
    }else{
        return 0;
    }
}
```