# REPORTE PSETS Y LABS

**Pset 0**: BlackPink Light Catch

Debes atrapar tantos lightstick de blackpink (negros y rosas) como puedas. ¡Ten cuidado de tocar los de Twice (amarillos y rosas)! Perderás una valiosa vida.

La caída de los lighsticks es controlada por las posiciones x & y del plano. Seleccionarán una posición random a lo largo de todo el juego, así también es asignado una velocidad por nivel. El sprite se moverá solamente de izquierda a derecha. Cada que toque un lightstick rosa y negro se sumará un punto. Si toca uno amarillo y rosa se restará una vida hasta que se acaben. Tiene un total de 4 niveles.

**Pset 1:**

* Hello:

Implementando cs50.h con la función get\_string es pedido su nombre al usuario, seguido de imprimirlo junto con un saludo. “Hello, Cinthia!”

* Mario-more:

Es pedido al usuario la altura de la pirámide validando que se encuentre dentro del rango [1,8]. Se usa el ciclo for para iterar en el rango de la altura ingresado, así mismo serán anidados 3 ciclos for más: El primero para imprimir los espacios, el segundo para imprimir hashes y el tercero para duplicar la pirámide. Por último, es impreso un salto de línea.

* Credit:

Definiendo una variable de tipo long donde se guardará el número de tarjeta del usuario. Es contada la longitud del número con ayuda de un ciclo while.

Si el contador es diferente a 13 y 15 y 16 entonces es inválida.

Luego se utiliza el algoritmo de Luhn para verificar que tipo de tarjeta es. Por último, se imprime la tarjeta y si no pertenece a ninguna de las existentes es inválida.

**Pset 2:**

* Readability:

Si el texto ingresado es parte del alfabeto se cuenta como letra, si no es omitido.

Si hay un espacio dentro de la longitud del texto se cuenta como una palabra más.

Si hay un punto, un signo de interrogación o admiración, es una oración más.

Dependiendo de estos 3 puntos, se hace el cálculo para indicar a que grado pertenece el texto.

* Caesar:

Primero se asegura el uso correcto de la llave. Si los argumentos son diferentes a 2 o si la llave es diferente a un dígito. Luego se convierte la llave a un entero.

El proceso de encriptación comenzará siempre y cuando el texto sea un alfabeto. Si es mayúscula es encriptado en dependencia de su código ASCII, así mismo si es minúscula. De otra manera será impreso tal cual.

**Pset 3:**

* Plurality:

*Vote:* Con un ciclo for puedes iterar sobre la cantidad de candidatos. Si el nombre dado está en la lista de candidatos entonces se le asignará un voto.

*Print\_winner:* Se verifica la mayor cantidad de votos. Si los votos del candidato son igual a la mayor cantidad entonces se imprime el nombre del candidato.

* Runoff:

*Vote:* Si el candidato se encuentra en las opciones entonces será guardado en las preferencias del votante y su rank de favoritos.

*Tabulate:* Recorres la cantidad de votos y de candidatos y verificas si el candidato no está eliminado. Si es así entonces los votos según la preferencia del votante aumentarán.

*Print\_winner:* Si el candidato tiene más de la mitad de votos de preferencia entonces será el ganador.

*find\_min:* Si el candidato no ha sido eliminado entonces se asigna a MAX\_VOTER la cantidad de sus votos.

*is\_tie:* Si el candidato no ha sido eliminado y el número de votos del candidato es distinto al mínimo retorna false.

*eliminate:* Si los votos del candidato son iguales a min entonces está eliminado.

**Pset 4:**

* Filter-less

Grayscale: Usando un ciclo for se recorre la altura y anchura de la imagen. Se calcula el promedio del valor del pixel y se asigna a los colores RGB de la imagen.

Sepia: Usando la fórmula para los colores sepia se compara si el resultado es mayor a 255. Si no es así significa que el pixel no tiene filtro y se asigna el valor de la fórmula.

Reflect: Guarda la imagen original temporariamente en un variable de tipo RGBTRIPLE y luego refleja la imagen derecha a la izquierda invertidamente.

Blur: Crea una imagen temporaria y recorre los pixeles verticalmente y horizontalmente, verifica si son válidos y están dentro de las filas y columnas y almacena el valor de ellos en una variable. Luego se redondean y asignan a la imagen temporaria.

Por último, recorre la original y la reemplaza por la temporaria.

* Recover

Es necesario primero validar la entrada de argumentos, si es diferente a 2 entonces cortará exitosamente el programa. Luego se abre un archivo tipo FILE en modo de lectura.

Definimos una array buffer de tipo unsigned char con valor de 512 bytes, otro archivo tipo FILE que tendrá como valor definido NULL y otro array tipo char.

Dentro de un ciclo while se lee los 512 bytes en el buffer. Si se encuentra el inicio de un nuevo JPEG, se determina que hay una imagen y se cierra el archivo. Luego se imprime el JPEG en el archivo nuevo en forma de 001.pg, 002.jpg y así sucesivamente.

Verifica si en realidad se impreso una imagen y la sobrescribe en el buffer.

Por útimo cierra todos los archivos que previamente fueron abiertos.

**Pset 5:** Speller

hash: Esta función recibe como argumento la palabra que se buscó y mediante y un ciclo for guarda en un arreglo un código hash de la palabra.

check: Recibirá como argumento la palabra del diccionario y verificará si se encuentra en él. Guardamos en una variable entera lo que retorna nuestra función hash y creamos un nodo que contendrá la lista donde se almacena el hash de la palabra.

Mientras que el nodo que definimos sea diferente a NULL, verificamos que la palabra sea igual al nodo que apunta a ella. Luego al nodo se le asignará el valor de él mismo, pero ahora apuntando al siguiente.

Load: Recibe como argumento el diccionario y se abre en un nuevo archivo de tipo FILE en modo de lectura.

Con un ciclo while leemos carácter por carácter el diccionario hasta que se acabe. Declaramos un nuevo nodo que tendrá un espacio de memoria dinámico de un nodo. Luego se copia la palabra en el nuevo nodo. Este mismo apuntará al nodo siguiente y este contendrá el valor hash de la palabra, así recíprocamente el índice donde se encuentra el hash en la lista será asignado el valor del nodo mientras su tamaño se va aumentando. Cerramos todos los archivos que anteriormente abrimos.

Size: Retorna el tamaño del diccionario que en la función load aumentó.

Unload: Recorre el número de espacios en el hashtable. Define un nodo que contendrá el valor de la lista en el índice correspondiente. Mientras que el nodo exista se creará un nodo temporal que guardará el valor del nodo. Luego el nodo original apuntará al siguiente. Por último, liberamos el nodo temporal.

Si el nodo ya no existe entonces el diccionario ya estará descargado de la memoria.

**Lab1:** Population

Pide al usuario el tamaño inicial y final de la población. Mientras el tamaño inicial sea menor al final aplicamos los cálculos correspondientes, así mismo los años aumentarán.

Por último, imprime el número de años que tomará.

**Lab2:** Scrabble

Compute\_score: Recorre la palabra y verifica si es mayúscula. Si esto es cierto, busca la letra en el arreglo de capitales con su código ASCII. Asigna puntos en un arreglo temporario y lo suma al puntaje. Si es minúscula hace lo mismo, esta vez con el arreglo de letras pequeñas.

Si no es alfabeto lo ignora.

Main: Si el puntaje del jugador 1 es mayor al del jugador 2 lo imprime como ganador y viceversa. Si no es así entonces es un empate.

**Lab3:** Sort

Ejecutando la palabra time antes del tipo de ordenamiento y los números podemos medir el tiempo de cada uno. Siendo así, el primer ordenamiento es de burbuja (Bubble Sort) ya que es el que más tiempo se tomó. El segundo es de mezcla (Merge Sort) ya que es el más rápido de los tres. El tercero es selección (Selection Sort), no es tan rápido como el de mezcla, pero sin duda más que el de burbuja.

Lab4: Volume

Copiamos el header del archivo de la entrada al de salida con el tipo uint8\_t y las funciones fread y fwrite. Definimos un buffer de tipo int16\_t y lo leemos con ayuda de un ciclo while y la función fread. Aplicamos el factor que contendrá la entrada de argumentos que proporcionó el usuario al buffer y lo sobrescribimos en el archivo de salida.

Por último, cerramos los archivos.

Lab5: Inheritance

\*create\_family: Alocamos memoria dinámicamente para una nueva persona de tipo person. Si hay más de una generación entonces la nueva persona apuntará a los padres en determinada posición. A este se le asignará el resultado recursivo de la función create\_family.

Asigna alelos al azar a la nueva persona basado en los alelos de los padres. Si no hay ptra generación entonces establecemos los punteros de los padres a NULL y asignamos alelos al azar a los hijos que hayan.

free\_family: usamos recursividad para liberar los padres de la persona y a la persona con la función free.

*Cinthia Carolina Duarte Ruiz.*