Design document for Assignment 3

Files:

a3.c – Contains the main function, and multithreading for the typeSearch() and saveResults() functions also contained

listFunctions.c – Contains the functions used for the ListType struct. addStringToList(), printList(), freeList()

defs.h – Defines structs, and constants, includes necessary header files, and contains forward declarations of all functions

pokemon.csv – The data file

Makefile – Allows for ease of file compilation and linking

Program Design:

My program design is based around three typedef struct definitions I have created:

```
typedef struct Node {
  char *data;
  struct Node *prev;
  struct Node *next;
} NodeType;

typedef struct {
  NodeType *head;
  NodeType *tail;
  int size;
} ListType;

typedef struct {
  ListType *pokemon;
  char *fileName;
  char *s;
} ParameterType;
```

The first two make up the basis of my linked list data structure that holds strings as the data type. I considered creating a pokemon struct in which each of the pokemon data is saved as a member of the struct, however I realized this was unnecessary as I only needed to process one of piece of data and then store the whole string onto memory. The third struct ParameterType is how I handled my multithreaded functions: to create a snappy and responsive program I used multithreading and created threads to complete search queries and while writing to the disk. The

reason why I needed a ParameterType struct was because the pthread_create() function only takes a single void * parameter and my threaded functions needed at least two parameters. The ParameterType allowed me to pass one parameter into pthread_create() while still giving my typeSearch() and saveResults() functions all the information they need. I used my ListType struct to store the pokemon data in memory and also to store the names of the new files created and since this data needed to be accessed by multiple threads I needed to use a mutex semaphore to prevent corruption. I simply locked the mutex at the start of my typeSearch() and saveResults() functions and unlocked at the end of them. To implement the requirements I needed to use both threads and semaphores so I made sure to include the <semaphore.h> header file and also compile with the lpthread library.

PSEUDO CODE OF A3.C:

```
volatile int numQueries = 0
main() {
        new threads t1, t2
        ListType pokemon, newFiles
        initList(pokemon, newFiles)
        scan(filename)
        try to open file and send error if file could not be opened
        while (user does not input 3)
                show options (query() function)
                if (user input 1)
                        search = prompt user to search for a pokemon type
                        ParameterType p = pokemon, filename, search
                        create thread (t1, typeSearch(), p)
                else if (user input 2)
                        newFile = prompt user to enter new file name
                        ParameterType p = pokemon, newFile
                        create thread (t2, saveResult(), p)
                        addStringToList(newFiles, newFile)
        print (Number of successful queries = numQueries)
        print(New files created:)
        printList(newFiles)
        freeList(newFiles)
        freeList(pokemon)
}
```

```
typeSearch(void *p) {
	lock semaphore mutex
	ParameterType parameters = (cast p to ParameterType)
	ListType pokemon = take from parameters
	char* filename = take from parameters
	char* search = take from parameters
	open(filename)
	while(read next line)
		token = everything until first comma
		token = first comma to second comma
		token = second comma to third comma (now token should equal to TYPE 1)
		if (compare(search, token) == 0)
			char* match = currentLine
			addStringToList(pokemon, match)
			count++
	if (count > 0)
		numQueries += 1
	unlock semaphore mutex
}

saveResults(void *p) {
	lock semaphore mutex
	ParameterType parameters = (cast p to ParameterType)
	ListType pokemon = take from parameters
	char* filename = take from parameters

	FILE out = open(filename, “for writing”)
	currentNode = pokemon.head
	while (currentNode is not the last node)
		write(out, currentNode.data)
		currentNode = next node
	close(out)
	unlock semaphore mutex
}
```