**Lab 5a: The ASCII Lab**

So far we have read in integers from the keyboard to our program using a nifty thing called a Scanner. This is something Java provides and is not in other languages. I’ve talked about what really happens there in class but thought a lab about this would be more useful. Back in the 1960’s all the people in the world got together and decided that for input to a computer (through a keyboard) and output to a screen (a video monitor …think about old T.V. sets if you can) would be **standardized**. This means that they wanted it so that if you typed a particular character on the keyboard that a specific thing would get sent to the computer. That specific thing would be a **number for that particular character.** “Why not just send the character”? …you ask. Well, you essentially are doing that … we’ve just mapped characters to numbers (because everything inside the computer is a number.

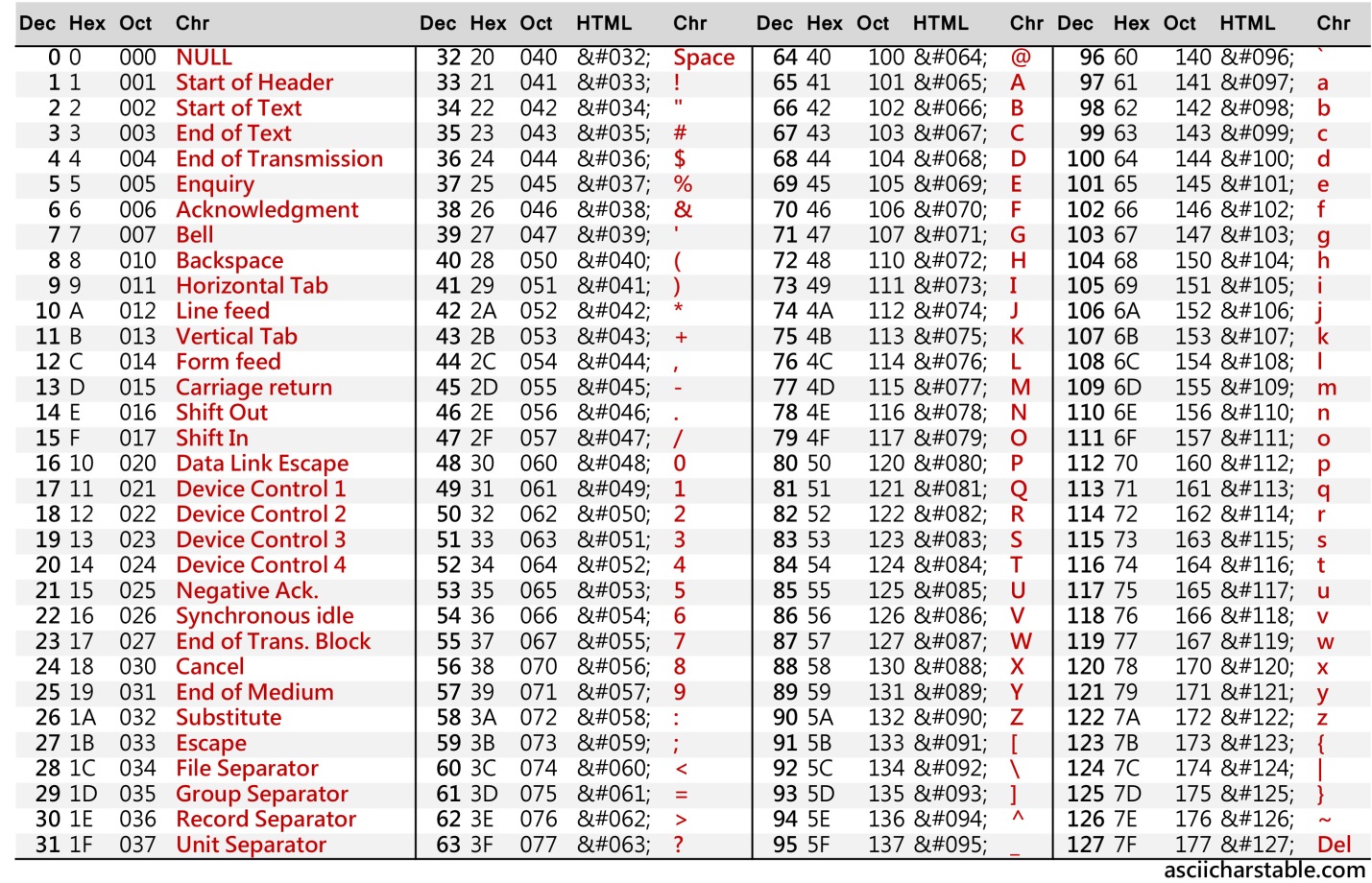
So, they created a thing called the ASCII code which stands for American Standard Code for Information Interchange (yeah, stupid name). This means that any program wishing to do I/O has to know the number code for a letter typed on the keyboard and also the number code when it wants to write a letter to the screen.

This code is pretty simple and I show it below. The big deal is though, nowadays we have a lot more characters than you can see on a keyboard. So, we now have an agreed on Universal Character Code. This comprises the characters from just about any language (think Cyrillic alphabet, Chinese characters etc.)

But, for almost all your applications you will use good old ASCII code because we use good old keyboards!

What is it? You can find it anywhere on-line. Here’s one example on the next page. It looks complicated. It isn’t. What you are seeing there is a character and then different number systems representation of that number. Why? Because if you write a lot of code you tend to think in terms of binary, octal, or hexadecimal representations of numbers … I know, we’re weird. So, if you can find the capital letter “A” on the chart, you will see that its’ code is the decimal (meaning the number system we normally count in) 65.

This means that if you type the letter “A” on the keyboard, the number 65 gets sent to your program when your program reads it. Looking at the table you notice that a “space” is the number 32. Get a feel for the table. It is very useful that numbers are consecutive and letters are consecutive. It makes it real easy to do certain things knowing that (this comes up in the lab). Also notice that the first column is a hodgepodge of strange things including a bunch of (literally) unprintable stuff. Also notice that the number 7 maps to the bell. When you print a “character 7” you make the little bell in your computer go off and it beeps at you. All pretty cool.



Now, how do you actually read something in? Well, here’s a very simple program I wrote to do this. Notice that after that magic “static void main” bit we’ve added the little phrase “throws IOException”. This is one of the weird things in Java. For now, just make sure you have this if you plan on reading from the keyboard.

import java.io.\*;

class charread {

public static void main(String args[]) throws IOException {

int charval = 0;

char mychar;

System.out.println("Please enter a character");

charval = System.in.read();

mychar = (char) charval;

System.out.println("Letter value read was " + charval + " character is "+ mychar);

} //main

}//charread

So, what is happening? When we execute the System.in.read() statement, it reads the next character but gets transmitted to the program as an integer. I assign what gets read in to the variable charval whose type is integer. The next statement converts that integer to a character. Any language derived from the language “C” supports this. This is called a “cast”. We are “casting” an integer to a character.

This makes the compiler happy because we are assigning to a character type. Study this program till you are sure you understand it. Run it and try it out by comparing what gets printed with what is in the ASCII table. Then (and only then) do the programs below.

Program 1: Have the above program that I created for you in your larger program. I should be able to give you characters from the keyboard (repeatedly without re-invoking the program) and you should tell me what their ASCII code is.  
  
  
Program 2: I should be able to give you a start character and an end character. You should then be able to print all the characters and ASCII codes (decimal only) between those two characters (inclusive). Again, repeatedly.  
  
Program 3: Replicate the ability of the nextInt function that you’ve been using in all the other programs. You don’t need to worry about the user typing in something bad. Just assume the user will give you up to a 6 digit integer followed by a space. You then take in each character using the System.in.read() stuff above and construct a number. Here’s one way to do this.

Say the user types in 923 followed by a space. You know the character 9 comes in as 57 decimal and you want it to store 9 in your “sum”. Well, all you need to do is subtract off the decimal representation of the number 0 (which is ASCII 48). In other words:

sum = numreadin – (int) ‘0’;

What did I do there? Let’s say I read in the character 9 and assigned it to numreadin (a variable). Well, the number 57 got shoved into that box. Then, I took the character 0 which in the program is delineated by tick marks (‘). This tells the compiler we are talking about a single character (as opposed to a whole string where we would use quotes. Now, I do that cool trick of “casting” and convert the character ‘0’ to a number. What happens? The thing to the right of the minus sign magically becomes 48 (that’s the ASCII code!! Now, we have 57 – 48 which is the NUMBER 9 and that gets shoved into sum. We’ve just done a conversion. All you have to do is figure out how to build a little loop around this to create the whole integer. Remember what you did to deconstruct a five-digit number and print it out in lab 2? You’re doing the reverse. When you’ve done this, you have just done 90% of what that in.nextInt thing you have does!

**What your program must do…repeatedly:**

I give you two numbers as a sequence of characters separated by a space and you give me their sum.

Example: “Please enter two numbers separated by a space. Each number should be 6 or fewer digits”

923 41

The sum is: 964

Just as an aside, the fact that these things are all numbers is what allows computers to sort lists and words etc. Pretty cool.