Caleb Hammond, Matthew Lockard

Requirment based design

sudo code design

```
int num1;
int num2;


"Please enter two numbers"

Scanner scan = new Scanner(); // intialize scanner
scan(); // take in two numbers from user

optional(make sure the numbers are +)

if(number must be pos);
{
 // do code;
}


Add(num1 + num2); // add the two numbers

int sum = Add(num1 + num2); //store the added number in a variable.

parse() // parse to get last digit
```

one option to get last digit is taking the mod of 10.

int last variable = sum % 10; store last number in a variable. // Parsed digit

Add(parse digit + original); // add the last digit of the number to the original digit.

int single = Add(parse digit + original);

//find a way to stop program when user gets a single number.

// when you get a single digit stop.

take two number in from user.

add the two numbers together.

parse the last digit to separate it from the added digit.

goal is to get a single digit.

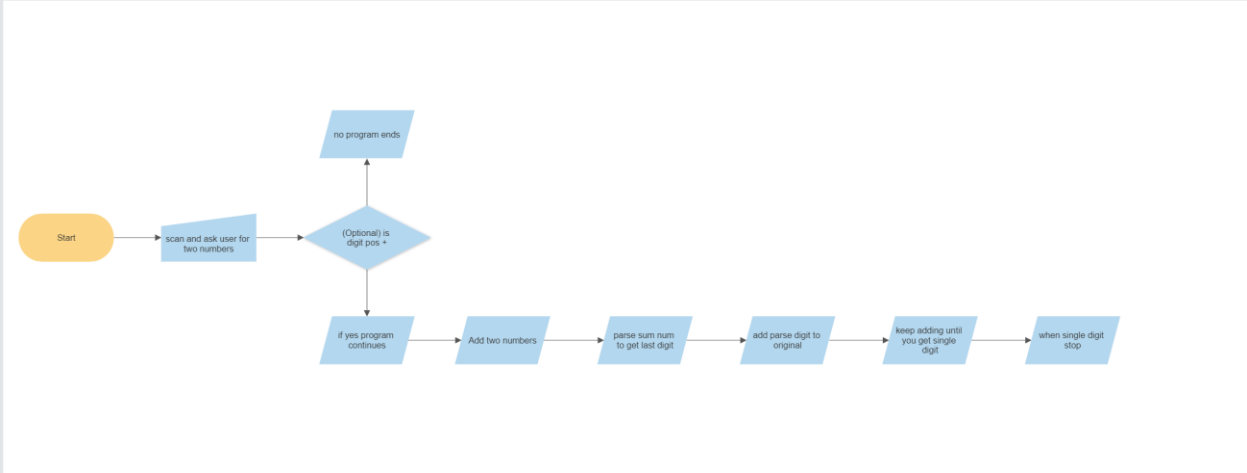
Take in a string

scan to ask user for a string.

parse string into set of characters and spaces.

loop through string to find the most repeated characters.

Flow chat



Take in a string (from a user or a file), compute the total number of occurrences of each character in the ASCII set. Print out the top three characters (with the most occurrences). Print out how many characters in the ASCII set you didn't detect. Then print out the string in reverse order.

The goal is to use a Requirements based design and use Pseudocode to get the requirements and logic down in English.

Requirements: first iteration

- Take in a string from a user or file
- Parse string to single chars
- Convert string/file to ASCII
- Print out the too three Chars (with the most occurrences
- Print out the remaining Chars in ASCII that you didn't detect
- Then print out the string in reverse order.

//We want to take an input from a user and store it as a string

// We are doing this because it's a hard requirement

Input: String from user

//Assign the user input to a variable UserInput, this allows use to manipulate the data saved

Assign input to variable: UserInput

//Inorder to convert the users string to ascii we must parse the entire string into individual chars so that we can then turn each char into its ascii equivalent

Parse: input into individual chars and store into an array named UserInputParsed as strings

//This for loop will parse each individual char from the parsing function into its ascii equivalent and then store each ascii char into its own array. From then we can scan the array to give us the top three most occurring ascii values aswell as give us the remaining values and reversed ordered output.

For (lets I = 0; I <UserInputParsed.length; i++) {

//Typecast is a built in feature to change a data type into a different data type

//Im storing each typecasted userinput into a variable that will be appended to another array

//called InputAsAsciiList from there we can parse that array to get our answers

InputAsAscii = Typecast(UserInputParsed[i]

InputAsAsciiList = InputAsAscii

}

Parse InputAsAsciiList will count each time an ascii char shows up and assign each char an int representing each time it shows up in the array. Then it will show the top three occurrences not just the Ascii characters. For example If their was a 4 way tie for the most occurrences, Lets say Ascii char 50,51,52,53 all occurs 10 times, then the output would show that the first most occurrences of ascii chars is 50,51,52,53 10 times. Following that if 54,55 showed up 9 times as the second most occurred chars, then the output would show 54,55 are the second most occurrences of ascii chars, then finally if 56 showed up 8 times as the third most occurrences for ascii chars, then the output would display 56 is the third most occurred char with 8 occurrences.

After determining the top three most common occurrences it will display the total number of ascii chars not considered for the top 3 occurred chars.

And then print(reverse(UserInput));

EX. First most occurred chars:

50,51,53,53

10 times

54.55

9 times

56

8 times

459 other chars not considered for top occurrences

(this represents then user input in reverse order)