COMP1911 22T2 (https://webcms3.cse.unsw.edu.au/COMP1911/22T2)

Code Examples from Lectures on 10-7_characters_and_strings

Introduction to Programming (https://webcms3.cse.unsw.edu.au/COMP1911/22T2)

asciiDemo.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/asciiDemo.c)

demonstrating how to print a character demonstrating the underlying ascii codes and their patterns.
DO NOT memorise ascii codes. We are only looking at them to understand how characters are actually stored.

ALWAYS use single quotes not raw numbers eg use 'A' NOT 65

Because there are patterns and orderings in the ascii table we can do maths and do things like 'a' + 1 and get 'b'

```
#include <stdio.h>


int main(void){

    printf("%d %c\n",65,65);  //This works but is BAD STYLE! ALWAYS USE 'A' etc
                              //Never use ascii codes directly


    printf("%d %c\n",'A','A');
    printf("%d %c\n",'a','a');

    printf("%d %c\n",'B','B');
    printf("%d %c\n",'b','b');


    printf("%d %c\n",'C','C');
    printf("%d %c\n",'c','c');

    printf("%d %c\n",'0','0');
    printf("%d %c\n",'7','7');

    //What would this give us?
    printf("%d %c\n",'A' + 25,'A' + 25);

    //What would this give us?
    printf("%d %c\n",'b' - ('a' - 'A'),'b' - ('a' - 'A'));

    //What would this give us?
    printf("%d %c\n",'Z' + ('a' - 'A'),'Z' + ('a' - 'A'));
    return 0;
}
```

printAscii.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/printAscii.c)

Print the 128 ASCII character encodings

```
#include <stdio.h>

#define ASCII_ENCODINGS 128


int main(void) {
    int i = 0;

    while(i < ASCII_ENCODINGS){
        printf("%d is %c\n",i,i);
        i = i + 1;
    }
    return 0;
}
```

charFunctions.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/charFunctions.c)

demonstrating how to write some functions with characters and use some of the char functions from ctype.h
Angela Finlayson

```c
#include <stdio.h>
#include <ctype.h>

// there is an equivalent function in ctype.h called toupper
//Converts a lower case letter to an upper case letter
//leaves other characters unchanged
int myToUpper(int c);

//prints lower case alphabet on a single line
void printAlphabet(void);


int main(void){
    int letter1 = 'z';
    int letter2 = 'A';
    int whatever = '?';

    printf("%c %c\n",letter1,
                   toupper(letter1));
    printf("%c %c\n",letter2,
                   myToUpper(letter2));
    printf("%c %c\n",whatever,
                   myToUpper(whatever));
    printAlphabet();
    return 0;
}

//Converts a lower case letter to an upper case letter
//leaves other characters unchanged
int myToUpper(int c){
   int result = c;
   // Could use function from ctype, islower
   // if(islower(c))
   if( c >= 'a' && c <= 'z') {
       result =  c - ('a' - 'A');
   }
   return result;
}

//prints lower case alphabet on a single line
void printAlphabet(void){
   int letter = 'a';
   while ( letter <= 'z'){
       printf("%c ",letter);
       letter = letter + 1;
   }
}
```

digits.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/digits.c)

Demonstrating how to conver a character digit into an integer

Angela Finlayson

Check through the code below and see if it prints out what you expect.

```c
#include <stdio.h>

//Convert a digit represented in ascii code, to its integer value
int digitToInt(int c);


int main(void){
    int number1 = 0;
    int number2 = 9;
    int digit1 = '0';  //Stored as 48
    int digit2 = '9';  //Stored as 57
    int result = digit1 + digit2;

    printf("%d + %d = %d\n",digit1,digit2,result);
    printf("%c + %c = %d\n", digit1, digit2, result);

    result = number1 + number2;
    printf("%d + %d = %d\n", number1, number2, result);

     //       0                 + 9
    result = digitToInt(digit1) + digitToInt(digit2);
    printf("%c + %c = %d\n", digit1, digit2, result);
    return 0;
}

// Convert a digit represented in ascii code, to its integer value
// For example if c is '9' (57) we return 9
int digitToInt(int c){
   if ( c >= '0' && c <= '9'){
       return c - '0';
   }
   return c;
}
```

getchar1.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/getchar1.c)

Reading and writing characters

```c
#include <stdio.h>

int main(void) {
    int c;
    printf("Please enter a character: ");
    c = getchar();
    printf("%c",c);
    putchar(c);
    putchar('\n');

    printf("The ASCII code of the %c is %d\n", c, c);

    return 0;
}
```

getchar2.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/getchar2.c)

getchar reads in newline characters and whitespace sometimes we want to read them in and discard them

Angela Finlayson

```c
#include <stdio.h>

int main(void) {
    int c1,c2;

    printf("Please enter first character:\n");
    c1 = getchar();
    getchar(); //discard the newline
    printf("Please enter second character:\n");
    c2 = getchar();
    printf("First %c\nSecond: %c\n", c1, c2);
    return 0;
}
```

getcharEOF.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/getcharEOF.c)

Read characters until eof

```
#include <stdio.h>

int main(void) {
    int ch;

    // getchar returns an int which will contain either
    // the ASCII code of the character read or EOF

    ch = getchar();
    while (ch != EOF) {
        printf("you entered the character: '%c'\n", ch);
        ch = getchar();
    }
    return 0;
}
```

getcharEOF2.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/getcharEOF2.c)

Read characters until eof

```
#include <stdio.h>

int main(void) {
    int ch;

    // getchar returns an int which will contain either
    // the ASCII code of the character read or EOF

    // using an assignment in a loop/if condition is
    // not recommended for noviceprogrammers
    // but is used widely by experienced C programmers

    while ((ch = getchar()) != EOF) {
        printf("you entered the character: '%c'\n", ch);
    }
    return 0;
}
```

classifyChar.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/classifyChar.c)

Revision Exercise:

Read in characters until we get to the EOF (press Ctrl ^d on a line of its own) and print out how many characters we read in overall, along with how many uppercase, lowercase, digits and other characters we read.

Angela Finlayson

```c
#include <stdio.h>
//#include <ctype.h> //if you want to use isupper,islower,isdigit

int main(void) {
   int c;
   int upper, lower, digits, other, total;

   total = 0;
   upper = 0;
   lower = 0;
   digits = 0;
   other = 0;

   c = getchar();
   while ( c != EOF ){
      total = total + 1;
      if ( c >= 'A' && c <= 'Z'){ //could use isupper(c)
          upper = upper + 1;
      } else if( c >= 'a' && c <= 'z'){ //could use islower(c)
          lower = lower + 1;
      } else if ( c >= '0' && c <= '9'){ //could use isdigit(c)
           digits = digits + 1;
      } else {
           other = other + 1;
      }
      c = getchar();
   }

   // print tally
   printf("Characters seen %d :\n ",total);
   printf("uppercase:\t%d\n lowercase:\t%d\n digits:\t%d\n other:\t\t%d\n",
         upper, lower, digits, other);

   return 0;
}
```

what0.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/what0.c)

Prints "My lecturer is awesome!\n" - using ASCII codes for the characters

Note: Please do not write programs like this one!!! Yuck! Very cryptic.

Compare the 8 what?.c programs which are all equivalent to get a better understanding of how & why C encodes character sequences

```c
#include <stdio.h>

int main(void) {
    int c1 = 77;  //M
    int c2 = 121; //y
    int c3 = 32;
    int c4 = 108;
    int c5 = 101;
    int c6 = 99;
    int c7 = 116;
    int c8 = 117;
    int c9 = 114;
    int c10 = 101;
    int c11 = 114;
    int c12 = 32;
    int c13 = 105;
    int c14 = 115;
    int c15 = 32;
    int c16 = 97;
    int c17 = 119;
    int c18 = 101;
    int c19 = 115;
    int c20 = 111;
    int c21 = 109;
    int c22 = 101;
    int c23 = 33;
    int c24 = 10;

    putchar(c1);   //equivalent to printf("%c",c1);
    putchar(c2);
    putchar(c3);
    putchar(c4);
    putchar(c5);
    putchar(c6);
    putchar(c7);
    putchar(c8);
    putchar(c9);
    putchar(c10);
    putchar(c11);
    putchar(c12);
    putchar(c13);
    putchar(c14);
    putchar(c15);
    putchar(c16);
    putchar(c17);
    putchar(c18);
    putchar(c19);
    putchar(c20);
    putchar(c21);
    putchar(c22);
    putchar(c23);
    putchar(c24);

    return 0;
}
```

what1.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/what1.c)

Prints "My lecturer is awesome!\n"

'M' is the C shorthand for the ASCII code for the character M (77)

Compare the 8 what?.c programs which are all equivalent to get a better understanding of how & why C encodes character sequences

```c
#include <stdio.h>

int main(void) {
    int c1 = 'M';
    int c2 = 'y';
    int c3 = ' ';
    int c4 = 'l';
    int c5 = 'e';
    int c6 = 'c';
    int c7 = 't';
    int c8 = 'u';
    int c9 = 'r';
    int c10 = 'e';
    int c11 = 'r';
    int c12 = ' ';
    int c13 = 'i';
    int c14 = 's';
    int c15 = ' ';
    int c16 = 'a';
    int c17 = 'w';
    int c18 = 'e';
    int c19 = 's';
    int c20 = 'o';
    int c21 = 'm';
    int c22 = 'e';
    int c23 = '!';
    int c24 = '\n';
    putchar(c1);
    putchar(c2);
    putchar(c3);
    putchar(c4);
    putchar(c5);
    putchar(c6);
    putchar(c7);
    putchar(c8);
    putchar(c9);
    putchar(c10);
    putchar(c11);
    putchar(c12);
    putchar(c13);
    putchar(c14);
    putchar(c15);
    putchar(c16);
    putchar(c17);
    putchar(c18);
    putchar(c19);
    putchar(c20);
    putchar(c21);
    putchar(c22);
    putchar(c23);
    putchar(c24);
    return 0;
}
```

what2.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/what2.c)

Prints "My lecturer is awesome!\n" using character constants to get the ASCII codes for the characters, store them in array, and print the array. Note we have to track the array length.

Compare the 8 what?.c programs which are all equivalent to get a better understanding of how & why C encodes character sequences

```
#include <stdio.h>

#define LENGTH 24

int main(void) {
    int asciiCodes[LENGTH];
    int i;

    asciiCodes[0] = 'M';
    asciiCodes[1] = 'y';
    asciiCodes[2] = ' ';
    asciiCodes[3] = 'l';
    asciiCodes[4] = 'e';
    asciiCodes[5] = 'c';
    asciiCodes[6] = 't';
    asciiCodes[7] = 'u';
    asciiCodes[8] = 'r';
    asciiCodes[9] = 'e';
    asciiCodes[10] = 'r';
    asciiCodes[11] = ' ';
    asciiCodes[12] = 'i';
    asciiCodes[13] = 's';
    asciiCodes[14] = ' ';
    asciiCodes[15] = 'a';
    asciiCodes[16] = 'w';
    asciiCodes[17] = 'e';
    asciiCodes[18] = 's';
    asciiCodes[19] = 'o';
    asciiCodes[20] = 'm';
    asciiCodes[21] = 'e';
    asciiCodes[22] = '!';
    asciiCodes[23] = '\n';


    i = 0;
    while (i < LENGTH) {
        putchar(asciiCodes[i]);
        i = i + 1;
    }

    return 0;
}
```

what3.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/what3.c)

Prints "My lecturer is awesome!\n"

using character constants to get the ASCII codes for the characters, initialize an array to the values , and print the array.

Note we still have to track the array length.

Compare the 8 what?.c programs which are all equivalent to get a better understanding of how & why C encodes character sequences

```
#include <stdio.h>

#define LENGTH 24

int main(void) {
    // if we don't specify the size of an array being initialized C will make
    // it big enough to hold all the initializing elements (14 in this case)
    int asciiCodes[] = {'M','y',' ','l','e','c','t','u','r','e','r',' ',
                        'i','s', ' ', 'a','w','e','s','o','m','e','!','\n'};
    int i;

    i = 0;
    while (i < LENGTH) {
        putchar(asciiCodes[i]);
        i = i + 1;
    }

    return 0;
}
```

what4.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/what4.c)

Prints "My lecturer is awesome!\n"

using character constants to get the ASCII codes for the characters, and initialize the array to the vales array using , and print the array.

This version has a extra special value in the array ('\0') to indicate the // end of the sequence. This means we no longer have to keep track of how many characters in the array (note the while loop condition)

Compare the 8 what?.c programs which are all equivalent to get a better understanding of how & why C encodes character sequences

```
#include <stdio.h>

int main(void) {
    // if we don't specify the size of an array being initialized C will make
    // it big enough to hold all the initializing elements
    int asciiCodes[] = {'M','y',' ','l','e','c','t','u','r','e','r',' ',
                        'i','s',' ','a','w','e','s','o','m','e','!','\n','\0'};
    int i;

    i = 0;
    while (asciiCodes[i] != '\0') {
        putchar(asciiCodes[i]);
        i = i + 1;
    }

    return 0;
}
```

what5.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/what5.c)

Prints "My lecturer is awesome!\n"

using character constants to get the ASCII codes for the characters, and initialize the array to the vales array using , and print the array.

This version has switched to using the numeric type char. This type is almost always 8 bits and shouldn't be used for arithmetic. It is commonly used to hold ASCII encodings.

Compare the 8 what?.c programs which are all equivalent to get a better understanding of how & why C encodes character sequences

```
#include <stdio.h>

int main(void) {
    // if we don't specify the size of an array being initialized C will make
    // it big enough to hold all the initializing elements
    char asciiCodes[] = {'M','y',' ','l','e','c','t','u','r','e','r',' ','i','s', ' ', 'a','w','e','s','o'
    int i;

    i = 0;
    while (asciiCodes[i] != '\0') {
        putchar(asciiCodes[i]);
        i = i + 1;
    }

    return 0;
}
```

what6.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/what6.c)

Prints "My lecturer is awesome!\n"

using character constants to get the ASCII codes for the characters, and initialize the array to the vales array using , and print the array.

C has a convenient shorthand for char arrays containing a sequence of

ASCII codes with an extra '\0' value marking the end of the sequence.

Compare the 8 what?.c programs which are all equivalent to get a better understanding of how & why C encodes character sequences

```
#include <stdio.h>

int main(void) {
    char asciiCodes[] = "My lecturer is awesome!\n";
    int i;

    i = 0;
    while (asciiCodes[i] != '\0') {
        putchar(asciiCodes[i]);
        i = i + 1;
    }

    return 0;
}
```

what7.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/what7.c)

Prints "My lecturer is awesome!\n"

using character constants to get the ASCII codes for the characters, and initialize the array to the vales array using , and print the array.

C has a convenient shorthand for char arrays containing a sequence of

ASCII codes with an extra '\0' value marking the end of the sequence.

A number of C library functions accept null-terminated char arrays

For example printf with the "%s" specification (below)

Compare the 8 what?.c programs which are all equivalent to get a better understanding of how & why C encodes character sequences

```
#include <stdio.h>

int main(void) {
    char asciiCodes[] = "My lecturer is awesome!\n";
    printf("%s", asciiCodes);
    return 0;
}
```

stringEx1.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/stringEx1.c)

C has a convenient shorthand for char arrays containing a sequence of

ASCII codes with an extra '\0' value marking the end of the sequence.

A number of C library functions accept zero-terminated char arrays

For example printf with the "%s" specification (below)

Try compiling and running with dcc and compare to gcc.

```
#include <stdio.h>

//What will this do?

int main(void) {


    //char message0[5] = {'H','e','l','l','o','\0'};
    char message1[6] = {'Z','e','l','l','o','\0'};
    char message2[] = {'J','e','l','l','o','\0'};
    //Not a string as no \0. Just an array of char
    char message3[] = {'H','e','l','l','o'};
    //C will add the \0 if we use ""
    char message4[] = "Hello";


    //printf("%s\n", message0);
    printf("%s\n", message1);
    printf("%s\n", message2);
    printf("%s\n", message3);
    printf("%s\n", message4);

    //We can print character by character too
    printf("%c\n",message2[2]);
    return 0;
}
```

fgetsDemo.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/fgetsDemo.c)

Simple example reading a line of input and examining characters

```
#include <stdio.h>
#include <string.h>

//#define SIZE 1024
#define SIZE 5

//Input to try
//dog
//bird
//hello
//newline
//ctrl^d

int main(void) {

    char input[SIZE];

    printf("Enter some input: ");
    if( fgets(input, SIZE, stdin) != NULL){
        printf("Input read %s\n",input);
    } else {
        printf("No input\n");
    }

    int len = strlen(input);
    printf("Len of string is %d\n",len);
    if( input[len-1] == '\n'){
       printf("I read in a whole line\n");
    } else {
        printf("I did not read in a whole line\n");
    }

    return 0;
}
```

fgetsEOF.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/fgetsEOF.c)

Read lines until eof

```c
#include <stdio.h>

//#define MAX_LINE 1024
#define MAX_LINE 10

int main(void) {
    char line[MAX_LINE];

    while(fgets(line,MAX_LINE,stdin) != NULL){
        printf("I read in %s\n",line);
    }

    return 0;
}
```

stringFunctions.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/stringFunctions.c)

Demonstrates how to use some of the functions from string.h and how we could write them ourselves

Angela Finlayson

```c
#include <string.h>
#include <stdio.h>



int myStrlen(char s[]);
void myStrCopy(char dest[], char src[]);

int main(void) {
    char str1[] = "hello";
    char str2[] = "goodbye";

    char str3[100];//has garbage at this stage

    //We can't assign arrays to each other
    //str3 = str1
    //We can use our myStrCopy(str3,str1);
    //Or the string.h funciton
    //strcpy(str3,str1);
    myStrCopy(str3,str1);

    printf("%s %s\n",str3,str1);

    printf("%d %d %d\n",myStrlen(str1),myStrlen(str2), myStrlen(str3));
    printf("%d %d %d\n",strlen(str1),strlen(str2), strlen(str3));

    //We can't compare two strings using ==. We use strcmp instead
    //if(str3 == str1){
    if (strcmp(str3,str1) == 0){
       printf("They are equal\n");
    } else {
        printf("They are not equal\n");
    }

    strcat(str3,str2);
    printf("%s\n",str3);
    return 0;
}
//"dog" 3 (array size 4)
//Our own version of the strlen function
//       i
// 0 1 2 3   s
// d o g \0
int myStrlen(char s[]){
   int i = 0;
   while(s[i] != '\0'){
       i = i + 1;
   }
   return i;
}

//       i
// 0 1 2 3   src
// d o g \0

// 0 1 2 3   dest
// d o g \0
//Our own version of the strcpy function
void myStrCopy(char dest[], char src[]){
   int i = 0;

   while(src[i] != '\0'){
       dest[i] = src[i];
       i = i + 1;
   }
   dest[i] = '\0';
}
```

search.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/search.c)

Check whether the user types in a hard-coded search key

```c
#include <stdio.h>
#include <string.h>

#define MAX_LINE 1024

int main(void){
    char searchKey[] = "hip hip array";
    char input[MAX_LINE];

    printf("Enter line : ");
    if(fgets(input,MAX_LINE,stdin) == NULL){
       printf("No data");
       return 0;
    }

    //remove trailing newline if there is one
    int len = strlen(input);
    if(input[len-1] == '\n'){
        input[len-1] = '\0';
    }

    //Ignore differences in the case (ie lower case and upper case) of letters
    if(strcasecmp(input,searchKey) == 0){
        printf("It is a match\n");
    } else {
        printf("It is not a match\n");
    }
    return 0;
}
```

arrayOfStrings.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/arrayOfStrings.c)

Example of storing arrays of strings

```c
#include <stdio.h>
#include <string.h>

#define MAX_LENGTH 20
#define  SIZE 10


//  0  1 2  3  4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
//0 d  o g  \0
//1 c  a t  \0
//2 e  l e  p  h a n t \0

void printAllStrings(char data[][MAX_LENGTH], int size);

int main(void){
    // dog rat tiger
    char names1 [][MAX_LENGTH] = {"dog","cat","elephant"};
    char names2 [][MAX_LENGTH] = {"dog","cat","bird","fish"};

    printAllStrings(names1,3);
    printAllStrings(names2,4);

    //How can we modify cat and change it to rat?
    names1[1][0] = 'r';
    printAllStrings(names1,3);
    strcpy(names1[2] ,"tiger");
    printAllStrings(names1,3);
    return 0;
}

void printAllStrings(char data[][MAX_LENGTH], int size){
    int i = 0;
    while ( i < size){
        printf("%s\n",data[i]);
        i = i + 1;
    }
}
```

fgets2dArray.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/fgets2dArray.c)

An example of reading into a 2darray

This assumes we have at most 10 4 letter words

```c
#include <stdio.h>

#define NUM_WORDS 10
//We have made it big enough to store the \n character as well
#define LEN 6

//  0 1 2 3 4 5
//0 b i r d \n \0
//1 t r e e \n \0
//2 d u c k \n \0
//3
//4
//etc
//9

//Read in strings into the array till we reach the EOF and return
//the number of strings we read in
int  readData(char data[NUM_WORDS][LEN]);
void printData(char data[NUM_WORDS][LEN], int numWords);

int main(void){
    int numWords = 0;
    //We can only store 10 strings
    //Each string can have a max length of 6
    //That means they can only store 5 characters because we need to store \0
    //If we also store the \n we can only really in
    //4 characters then the \n then the \0
    char data[NUM_WORDS][LEN];


    numWords = readData(data);
    printf("I read in %d words\n",numWords);

    printf("Printing out the words\n");
    printData(data,numWords);

    return 0;
}

//Read in strings into the array till we reach the EOF and return
//the number of strings we read in
int  readData(char data[NUM_WORDS][LEN]){
    int i = 0;
    while(i < NUM_WORDS && fgets(data[i],LEN,stdin) != NULL){
        i = i + 1;
    }
    return i;
}

void printData(char data[NUM_WORDS][LEN], int numWords){
    int i = 0;
    while ( i < numWords){
        printf("%s\n",data[i]);
        i = i + 1;
    }
}
```

printArguments.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/printArguments.c)

print command line arguments

```
#include <stdio.h>
#include <stdlib.h>

//YOU NEED argc and argv and not just void to use command
//line arguments


//./print_arguments hello 99

// argc = 3
// argv
// 0: "./print_arguments"
// 1: "hello"
// 2: "99"

//For now just think of argv as an array of strings
int main(int argc, char * argv[]) {
    int i;
    printf("argc is %d\n",argc);
    i = 0;
    while ( i < argc ){
        printf("%s\n",argv[i]);
        i = i + 1;
    }

    return 0;
}
```

calc.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/10-7_characters_and_strings/code/calc.c)

sum integers supplied as command line arguments

```
#include <stdio.h>
#include <stdlib.h>

//  0      1  2   3  .....
// ./calc 99 100 -10


// argc: 4
// argv
// 0: "./calc"
// 1: "99"
// 2: "100"
// 3: "-10"

int convertToInt(char s[]);

int main(int argc, char * argv[]) {
    int total  = 0;

    int i = 1;
    while ( i < argc ){
        total = total + atoi(argv[i]);
        i = i + 1;
    }

    printf("Total is %d\n",total);
    return 0;
}

//our own version of atoi
int convertToInt(char s[]){
    int num = 0;
    int sign = 1;
    int i = 0;

    if(s[0] == '-'){
        sign = -1;
        i = 1;
    }

    while(s[i] != '\0'){
        num = (num*10) + s[i] - '0';
        i = i + 1;
    }
    return num*sign;
}
```