SDSU | San Diego State University

Lab1 CS420

Eddie Coda
#817061330

Decimal to Binary Calculator

Summary

The assignment prompt was to create a program that would take a decimal number as input from the user, and then convert that number to binary, in addition we needed to calculate the respective 1's and 2's complement. The program also allows the user to choose the bit system they want to output the binary number in, from 4 to 64 bits.

To start, I created a main function that would allocate memory for the input decimal number, the bit system size, and the input validity flag. I used a do-while loop to prompt the user to input a valid decimal number, and then another do-while loop to prompt the user to input a valid bit system size.

Once the user inputs a valid decimal number and bit system size, I use a while loop to calculate the binary representation of the input decimal number. I used realloc to dynamically allocate memory for the binary output array, and then used mod and division to convert the decimal number to binary. After that, I performed the 1's and 2's complement on the binary output using for loops and conditional statements. I used malloc to dynamically allocate memory for the 1's and 2's complement arrays, and then used bitwise operations to perform the complement methods.

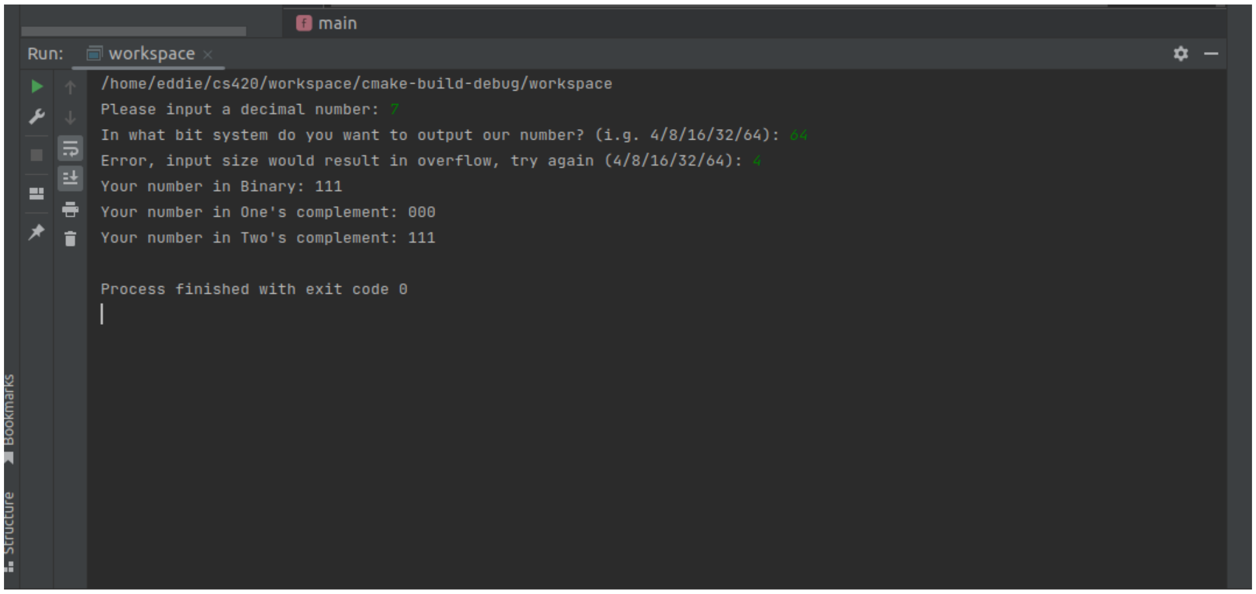
Finally, I printed out the binary value, 1's complement, and 2's complement representations of the input decimal number, and then freed the dynamically allocated memory.

This assignment was a bit confusing to me at first, I had a lot of issues with the sizes and the memory allocation/reallocation. It allowed me to practice allocating and freeing memory dynamically, as well as using loops and conditional statements to perform calculations. And, of course, it helped me to better understand pointers and their role in C programming.

Sample Code

Final Output:

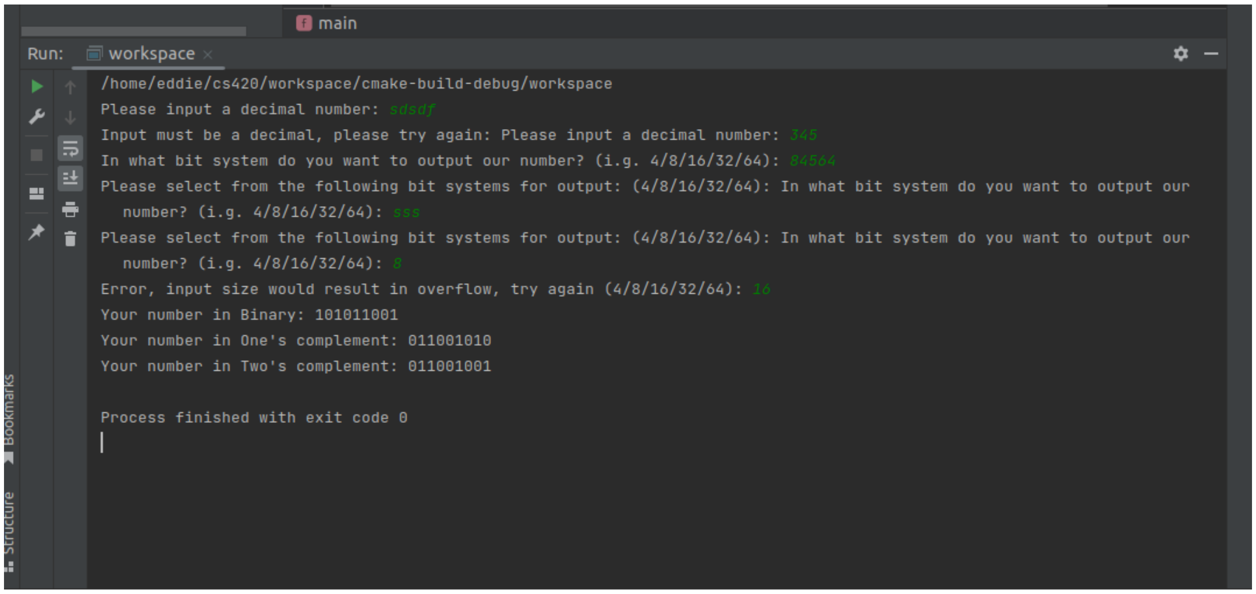
Here I have the output displaying the error checking for incorrect input size that would result in an overflow:



```
/home/eddie/cs420/workspace/cmake-build-debug/workspace
Please input a decimal number: 7
In what bit system do you want to output our number? (i.g. 4/8/16/32/64): 64
Error, input size would result in overflow, try again (4/8/16/32/64): 4
Your number in Binary: 111
Your number in One's complement: 000
Your number in Two's complement: 111

Process finished with exit code 0
```

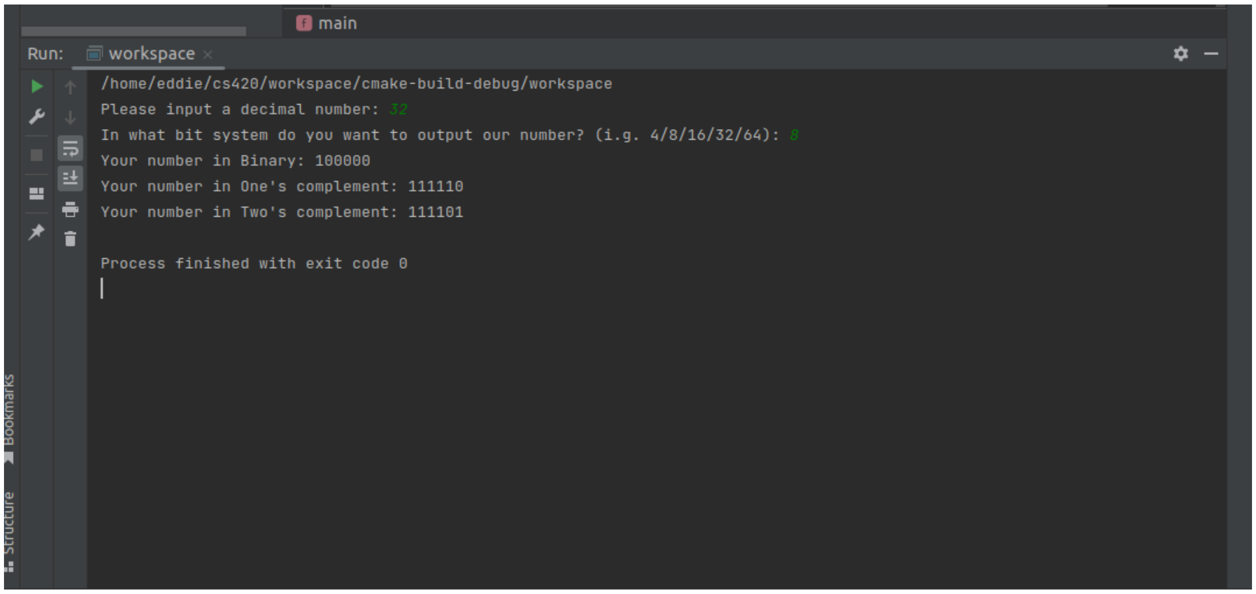
Here is the output when I try to enter random values and then finally enter valid values:



```
/home/eddie/cs420/workspace/cmake-build-debug/workspace
Please input a decimal number: sdsdf
Input must be a decimal, please try again: Please input a decimal number: 345
In what bit system do you want to output our number? (i.g. 4/8/16/32/64): 84564
Please select from the following bit systems for output: (4/8/16/32/64): In what bit system do you want to output our
  number? (i.g. 4/8/16/32/64): sss
Please select from the following bit systems for output: (4/8/16/32/64): In what bit system do you want to output our
  number? (i.g. 4/8/16/32/64): 8
Error, input size would result in overflow, try again (4/8/16/32/64): 16
Your number in Binary: 101011001
Your number in One's complement: 011001010
Your number in Two's complement: 011001001

Process finished with exit code 0
```

Here is the output for the dec input 32 to match the provided example by the Professor:



```
/home/eddie/cs420/workspace/cmake-build-debug/workspace
Please input a decimal number: 32
In what bit system do you want to output our number? (i.g. 4/8/16/32/64): 8
Your number in Binary: 100000
Your number in One's complement: 111110
Your number in Two's complement: 111101

Process finished with exit code 0
```

Other Documented Issues:

- memory issues are not letting me proceed on to calculate the 1's and 2's
- Using realloc could simplify code and make it more efficient?? Instead of allocating new memory in the copied binary output....use realloc to resize the binary output array and then perform the 1's and 2's
- "double free or corruption (out)" error
    - the program is attempting to free a block of memory that has already been freed???
    - created a separate copy of binaryoutput and used that copy in the calculation of two's complement, rather than using the original memory location
- Issues with code where I am only allocating a single integer for “binaryoutput” and needed to allocate an array with enough space.
- In the first attempt, the allocation for binaryoutput was only for a single integer, not an array of integers, which caused the memory issues. In the updated code, the allocation for binaryoutput is correct, but there seems like there is still issues in the while loop that generates the binary value.

- issue is that the array binaryoutput is initially allocated with size bitsize, which is the desired bit system for output, but the size of binaryoutput needs to be dynamic

Notes:

- **Start by initializing arrayStartPtr to point to the first element of binaryoutut, arrayEndPtr to point to the last element of binaryoutut, and resultStartPtr to point to the last element of twoscomplement.**
- **Then loop through the binaryoutut array,**
    - **starting from the end, and perform a bitwise XOR operation on each bit with carry,**
    - **and store the result in twoscomplement**
    - **then decrement the arrayEndPtr and resultStartPtr pointers,**
    - **and repeat the loop until we have processed all the bits in the binaryoutut array.**
- **eliminate the carry variable by using two pointers to keep track of the start and end of the twoscomplement array**
- **use three integer pointers: arrayEndPtr to iterate through the binaryoutut array from the end to the beginning, resultStartPtr to fill in the twoscomplement array from the end to the beginning, and resultEndPtr to fill in the twoscomplement array from the beginning to the end.**
- **start by initializing resultStartPtr to point to the last element of twoscomplement, resultEndPtr to point to the first element of twoscomplement, and arrayEndPtr to point to the last element of binaryoutut. Then loop through the binaryoutut array, starting from the end, and perform the one's complement of each bit, then perform the two's complement of each bit.**
- ***arrayEndPtr ^ (resultStartPtr != resultEndPtr), XORs the current bit of binaryoutut with a boolean value that is true if resultStartPtr != to resultEndPtr**

Source Code Copy:

```
/* Decimal to Binary Conversion and Two's Complement Representation
 * Author: Eddie Coda
 */
#include <stdio.h>
#include <stdlib.h>

int main() {
    /* Dynamic memory allocation */
    int* inum= (int*) malloc(sizeof(int));
```

```
int* bitsize = (int*) malloc(sizeof(int));
int* inputdec = (int*) malloc(sizeof(int));

/* Get Decimal input value */
do {
    printf("Please input a decimal number: ");
    *inputdec = scanf("%d", inum);

    if(*inputdec != 1){
        printf("Input must be a decimal, please try again: ");
        while(getchar() != '\n'); // clear input buffer
    }
} while(*inputdec != 1);

/* Get bit size from user */
do {
    printf("In what bit system do you want to output our number? (i.g. 4/8/16/32/64): ");
    scanf("%d", bitsize);

    if(*bitsize !=4 && *bitsize!=8 && *bitsize != 16 && *bitsize != 32 && *bitsize != 64){
        printf("Please select from the following bit systems for output: (4/8/16/32/64): ");
        while(getchar() != '\n'); // clear input buffer
    }

    if(*inum >= (1<< *bitsize)){
        printf("Error, input size would result in overflow, try again (4/8/16/32/64): ");
        scanf("%d", bitsize);
    }
} while(*bitsize !=4 && *bitsize!=8 && *bitsize != 16 && *bitsize != 32 && *bitsize != 64);

/* allocating memory for output ptr */
int* binaryoutput = (int*) malloc(sizeof(int) * (*bitsize));
int index = 0;

/* Dec to Bin */
while (*inum > 0) {
    binaryoutput[index++] = *inum % 2;
    *inum /= 2;
}
/* Reallocate memory for output ptr */
binaryoutput = (int*) realloc(binaryoutput, index * sizeof(int));

/* Perform 1's complement */
```

```
    int* onescomp = (int*) malloc(sizeof(int) * index);
    for (int i = 0; i < index; i++) {
        onescomp[i] = binaryoutput[i] == 0 ? 1 : 0;
    }

    /* Perform 2's complement */
    int* twoscomp = (int*) malloc(sizeof(int) * index);
    int carry = 1;
    for (int i = index - 1; i >= 0; i--) {
        twoscomp[i] = binaryoutput[i] ^ 1;
        if (carry) {
            twoscomp[i] ^= 1;
            carry = binaryoutput[i];
        }
    }

    /* Print binary, 1's, and 2's complement */
    printf("Your number in Binary: ");
    for (int i = index - 1; i >= 0; i--) {
        printf("%d", binaryoutput[i]);
    }
    printf("\n");

    printf("Your number in One's complement: ");
    for (int i = 0; i < index; i++) {
        printf("%d", onescomp[i]);
    }
    printf("\n");

    printf("Your number in Two's complement: ");
    for (int i = 0; i < index; i++) {
        printf("%d", twoscomp[i]);
    }
    printf("\n");

    /* Free memory */
    free(binaryoutput);
    free(onescomp);
    free(twoscomp);
    free(inum);
    free(bitsize);
    return 0;
}
```