APSC 160 MIDTERM 1 Review Package

UBC Engineering Undergraduate Society

Attempt questions to the best of your ability. This review package consists of 36 pages, including 1 cover page and 28 questions. The questions are meant to be the level of a real examination or slightly above, in order to prepare you for the real exam. Material from lectures and from the relevant textbook sections is examinable, and the problems for this package were chosen with that in mind, as well as considerations based on past examination question difficulty and style.

Solutions posted at: `https://ubcengineers.ca/review-packages` If you believe that there is an error in these solutions, or have any questions, comments, or suggestions regarding EUS Tutoring sessions, please e-mail us at: eus.ubc.tutoring@gmail.com. If you are interested in helping with EUS tutoring sessions in the future or other academic events run by the EUS, please e-mail eus.ubc.academic@gmail.com.

Some of the problems in this package were not created by the EUS. Those problems originated from one of the following sources:

- EUS
- APSC 160 Professors

All solutions prepared by the EUS.

Good Luck!

1 Multiple choice

1. What does a C language compiler do?
   - A. translate from C to machine language
   - B. verify correctness of a program
   - C. process directives like #include and #define before running the program
   - D. organize all the C program files in a directory

Solution: A

2. Which is the correct way to declare a variable s containing the string foo?
   - A. char s = 'foo';
   - B. char s = "foo";
   - C. char s[ ] = "foo";
   - D. char s[ ] = "foo"\0;

Solution: C

3. Which of the following expressions is always TRUE? Assume p is an int variable.
   - A. p | | !p
   - B. p && !p
   - C. !p | | !p
   - D. p && p

Solution: A

4. A global variable
   - A. is accessible to any function that wants to use it
   - B. is initialized every time a function is called
   - C. can not be changed
   - D. allows faster access than a normal variable

Solution: A

5. Character variables in C are passed to functions
   - A. by name
   - B. by reference
   - C. by value
   - D. as strings

Solution: C

6. Which is the correct way to declare a variable s containing the string foo?
   - A. char s = ‘foo’;
   - B. char s = “foo”;
   - C. char s[ ] = “foo”;
   - D. char s[ ] = foo;

Solution: C

7. What is the binary number 1011 in decimal form?
   - A. 23
   - B. 11
   - C. 5
   - D. 13

Solution: B

8. What is the decimal number 6 in binary form?
   - A. 1011
   - B. 111
   - C. 100
   - D. 110

Solution: D

9. What is the function of the statement ‘return 0;’
   - A. Stops the progression of all code
   - B. Terminates the execution of the function main
   - C. Terminates all functions in the code
   - D. Restarts the code

Solution: B

10. Which is the correct way cast a variable to a new type?
    - A. average = (float)sum/count
    - B. (float)average = sum/count
    - C. average = sum/count * float
    - D. average = sum(float)/count

Solution: A

2 Reading Code

11. Explain in plain English the purpose of the following function:

```
/*
* Param: data - an array of integers
* Param: numVals - the number of entries in the array
* Param: checkValue - a threshold value
*/

int procList( int data[ ], int numVals, int checkValue) {
    int count = 0;
    int index;

    for( index = 0; index $<$ numVals; index++) {
        if( data[ index ] $>$ checkValue )
            count++;
        }
    return count;
}
```

Solution: Returns the number of elements in the array that are greater than checkValue

12. Explain in plain English the purpose of the following function. Assume that TRUE, FALSE, and SIZE are symbolic constants defined with appropriate values.

```c
/*
* Param: data - a two dimensional array with SIZE rows
* and SIZE columns
* Param: nRows - a value representing a number of rows in
* the array data such that 1 $<=$ nRows $<=$ SIZE
*/

int checkArray(int data[SIZE][SIZE], int nRows) {
    int isSomething = TRUE;
    int row;

    for(row = 0; row < nRows; row++) {
        int col;

        for( col = row + 1; col < nRows; col++) {
            if( data[row][col] != data[col][row] )
                isSomething = FALSE;
        }
    }
return isSomething;
}
```

Solution: Returns true if the square array lying at the intersection of the first nRows rows and first nRows columns is symmetric and false otherwise. By symmetric we mean that if you reflect the array about the main diagonal (running from top-left to bottom-right corner), you get the same array.

3 Conditional statements (if / else if / else)

13. Create a program that will convert units. The options for conversions are:
   (a) km to mile
   (b) mile to km
   (c) kg to pound
   (d) pound to kg

   Your system must prompt the user for what kind of conversion they would like, and give an error message if they do not enter a correct option. Your system will run once only (no looping). You can look up the conversion factors to use in your code.

Solution:

```c
#define_CRT_SECURE_NO_WARNINGS

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

#define KM_MILES_CONVERSION 0.621371
```

```
#define KG_POUNDS_CONVERSION 2.20462

int main(void){
    int conversion_type;
    float initial_value;
    float final_value;

    //provide conversion options
    printf("These are the unit conversion options available:\n\n");
    printf("km to miles.......1\n miles to km.......2\n kg to pounds......3\n pounds
     to kg......4\n\n ");

    //enter value to be converted
    printf("Please enter the number of the conversion you would like to do: \n");
    scanf("\%d", \&conversion\_type);

    //confirm a correct conversion was selected

    if (1 <= conversion_type && conversion_type <= 4){
        printf("Please enter your initial value:\n");
        scanf("%f", &initial_value);

        //begin possible conversions
        if (conversion\_type == 1){
            final_value = initial_value * KM_MILES_CONVERSION;
        }
        else if (conversion_type == 2){
            final_value = initial_value / KM_MILES_CONVERSION;
        }
        else if (conversion\_type == 3){
            final_value = initial_value * KG_POUNDS_CONVERSION;
        }
        else if (conversion_type == 4){
            final_value = initial_value / KG_POUNDS_CONVERSION;
        }

        printf("Your converted value is %.4f\n\n", final_value);
    }

    //let user know if they have not selected a proper option
    else{
        printf("You must enter an integer from 1-4\n");
    }

    system("PAUSE");
    return 0;
}
```

4 Loops (for, while)

14. Complete the program provided below so that it prompts the user for a series of positive integers. The user enters the value -1 to indicate that there is no more data to enter. The function must print the smallest and largest of the integer values entered by the user (exluding the value -1). If the first value entered by the user is -1 (in other words, no positive integers are entered), your program must print an error message and terminate. Here are a couple of sample runs:

Sample run #1

```
Enter a series of positive integers,
terminated by -1:
-1
ERROR: no data entered!
Press any key to continue...
```

Sample run #2

```
Enter a series of positive integers,
terminated by -1:
4
2
10
3
-1
Minimum value: 2, maximum value: 10
Press any key to continue...
```

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

#define SENTINEL -1

int main( void ) {
```

Solution:

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

#define SENTINEL -1

int main(void){
    int next;
    int min;
    int max;

    printf( "Enter a series of positive integers,\n" );
    printf( "terminated by %d:\n", SENTINEL );
    scanf( "%d", &next );

    min = max = next;
    while( next != SENTINEL ) {
        if( next > max )
            max = next;
        else if( next < min )
            min = next;

        scanf( "%d", &next );
    }

    if( max == SENTINEL )
        printf( "ERROR: no data entered!\n" );
    else
        printf( "Minimum value: %d, maximum value: %d\n", min, max );

    system( "PAUSE" );
    return 0;
}
```

15. Write a program in C to read 10 numbers from keyboard and find their sum and average.

Solution:

```
#include <stdio.h>
int main void(){
    int i,n,sum=0;
    float avg;

    printf("Input the 10 numbers : \n");

    for (i=1; i$<=$10; i++){
        printf("Number-%d :",i);
        scanf("%d",&n);
        sum +=n;
    }

    avg=sum/10.0;

    printf("The sum of 10 numbers is : %d\n The Average is : %f\n",sum,avg);
}
```

5 Functions

16. Write a function in C that computes and returns (as a double value) the sum of the following series of numbers: 1 + x + x^2 + x^3 + x^4 + x^{N-1} + x^N where the integer N and the double x are passed to the function as parameters. Here are some examples:

N	x	value returned	explanation
0	3.3	1	always 1 for N=0
3	1	4	$1 + 1 + 1^2 + 1^3$
2	.5	1.75	$1 + 0.5 + 0.5^2$

Note that your function is not expected to print the explanation – this has been provided only to help you understand what the function is supposed to do. For full marks on this question, do not use the pow math library function or any function that you write yourself that computes one number raised to the power of another. Note that arrays are not needed in this question and must not be used.

Solution:

```
/*
* Param: x
* Param: n - there are n+1 terms in the series total (0 : n)
* Return: $1 + x + x^2 + ... + x^n$
*/

double computeSeries (double x, int n){
    int i;
    double xn = 1.0, result = 1.0;

    for(i = 1; i <= n; i++) {
        xn *= x;
        result += xn;
    }
    return result;
}
```

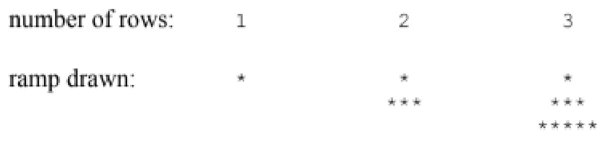
17. Assume that the following function has already been written and is available for you to use in this question:

```c
/*
* Draws a character a specified number of times in a row
* Param: numChars - the number of times to print the character
* Param: printChar - the character to print
*/

void printChars( int numChars, char printChar);
```

Wite a function named *drawRamp* that:

- takes a postive integer representing a number of rows as its *only* parameter
- draws a ramp of *'s

Your function must make use of the printChars function described above. Here are some examples:



```
number of rows:   1     2       3

ramp drawn:       *      *        *
                        ***      ***
                                *****
```

Solution:

```c
/*
* Draws a picture of a ramp having size rows
* Param: size - the size of the ramp
*/

void printRamp( int size ){
    int row;

    for( row = 1; row <= size; row++ ){
        printChars( size - row, ' ' );
        printChars( 2 * row - 1, '*' );
        printf( "\n" );
    }
}
```

18. In this question you will write a program in C that will allow a user to convert currencies. When the program starts up an introduction message is displayed which tells the user what the code numbers for the supported currencies are. The user is then prompted to input the currency type to convert from (the user enters an integer here), the amount to convert (the user enters the amount), and the currency type to convert to (also represented by an integer). When this data has been entered the program displays the converted amount with two decimal points accuracy. After the conversion the program asks the user if another conversion is wanted. If so the steps listed above (including the introduction message) are executed again. If not the program terminates Here is a sample run. User input is in **bold** type:

    > Welcome to CConvert 1.0.

    > 1=USD, 2=CAD,3=EUR,4=ITL,5=DEM,6=NLG,7=FRF,8=CHF

    > Enter currency to convert: **2**

    > Enter the amount to convert: **120.15**

    > Enter currency to convert to: **1**

    > 120.15 CA$ equals 106.46 US$

    > Do you wish to convert again? (Y/N): **Y**

    > 1=USD, 2=CAD, 3=EUR, 4=ITL, 5=DEM, 6=NLG, 7=FRF, 8=CHF

    > Enter currency to convert: **6**

    > Enter the amount to convert: **20.65**

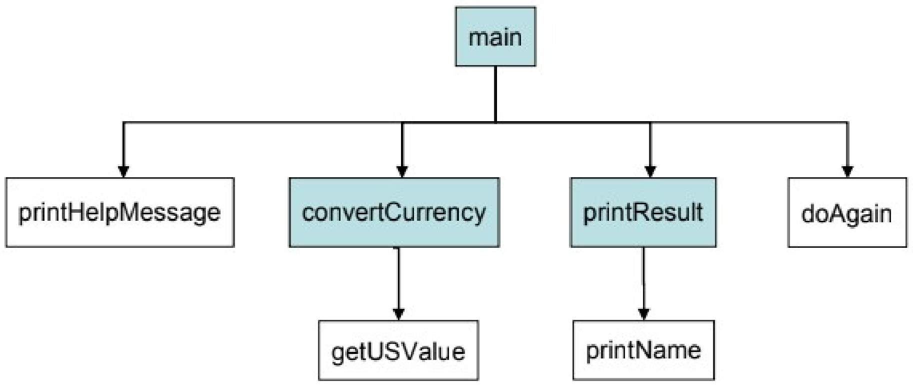
    > Enter currency to convert to:**5**

    > 20.65 Dutch Guilder equals 18.34 DM

    > Do you wish to convert again? (Y/N): **N**

    > Bye!

    The following diagram is a modular structure chart to represent the design of the program that you will write. Assume that the functions in white boxes have been given to you, you do not have to write them, just use them. A description of each function appears below in the prototype section of the code.


main
printHelpMessage
convertCurrency
printResult
doAgain
getUSValue
printName

In the questions that follow, you may assume that the following code has already been inserted at the top of the file:

```
#include <stdlib.h>
#include <stdio.h>
#define TRUE 1
#define FALSE 0

/* Given functions (you can use them, but don't have to write them) */

/* print to screen (without newline) the name of currency with code c
*/
void printName(int c);

/* prompt the user for Y or N, returns FALSE if user entered N, TRUE if Y
*/
int doAgain(void);

/* Get the US$ value of 1 unit of specified currency with code c
*/
double getUSValue(int c);

/* print to screen the help message with the currency codes. In the example
above this was:
1=USD, 2=CAD,3=EUR,4=ITL,5=DEM,6=NLG,7=FRF,8=CHF
*/
void printHelpMessage(void);

/* Functions you will have to write */

/* convert amount from currency fromCode to currency toCode and return
converted amount
*/
double convertCurrency(double amount,int fromCode,int toCode);

/* print to screen the result of the conversion in precisely the format
given in the example run above. For example "12.12 US$ equals 13.68 CA$"
*/
void printResult(double convertedAmount,double fromAmount,
                                int fromCode,int toCode);

/* main function and function implementations follow below this */
```

(a) Complete the missing parts of the function main. You don't have to check the user input for correctness.

```
int main(void) {
    int fromCode,toCode;
    double fromAmount,convertedAmount;
    printf("Welcome to CConvert 1.0.\n");

    do {









        convertedAmount = convertCurrency(fromAmount,fromCode,toCode);
        printResult(convertedAmount,fromAmount,fromCode,toCode);

    } while(                                             );

    printf("Bye!\n");
    return 0;
}
```

(b) Write the complete implementation of the function convertCurrency.

(c) Write the complete implementation of the function printResult. Hint: look at the modular structure chart to see if you can use one of the provided functions.

Solution:

(a)
```
int main(void){
    int fromCode,toCode;
    double fromAmount,convertedAmount;

    printf(  Welcome   to CConvert 1.0.\ n  );
    do{
        printHelpMessage();

        printf( "Enter currency to convert: " );
        scanf( "%i", &fromCode );

        printf( "Enter amount to convert: " );
        scanf( "%lf", &fromAmount );

        printf( "Enter currency to convert to: " );
        scanf( "%i",&toCode );

        convertedAmount = convertCurrency( fromAmount, fromCode, toCode );

        printResult( convertedAmount, fromAmount, fromCode, toCode );

        convertedAmount = convertCurrency(fromAmount,fromCode,toCode);
        printResult(convertedAmount,fromAmount,fromCode,toCode);
    } while( doAgain() );

    printf("Bye!\n");

    return 0;
}
```

(b)
```
double convertCurrency(double amount, int fromCode, int toCode) {

    return amount * getUSValue( fromCode ) / getUSValue( toCode );
}
```

(c)
```
void printResult(double convertedAmount, double fromAmount,
int fromCode, int toCode){

    printf( "%5.2f ", fromAmount );
    printName( fromCode );
    printf( " equals " );
    printf( "%5.2f ", convertedAmount );
    printName( toCode );
    printf( "\n" );
}
```

6 FILE I/O

19. A quality control engineer must ensure that ball bearings manufactured by her company conform to the advertised standards. She has a file called ballBearings.dat that contains information about a sample of ball bearings. Each row of the file contains data about one ball bearing. Data is organized into two columns of non-integer values separated by a space. The first column contains a diameter in mm and the second a weight in grams. There is no other data in the file. The number of lines of data in the file is not known but you can assume that there is at least one line of data.

A ball bearing is considered to be within accepted standards if (9.95 ¡ diameter ¡ 10.02), otherwise it is rejected. We want to write a program that reads the file ballBearings.dat described above and determines the percentage of ball bearings that meet accepted standards and the percentage that are rejected. The program must display these percentages on the screen to 1 decimal place along with appropriate labels.

The partially completed program appears below and on the following pages. To complete it you must implement the functions: doQualityControl and printReport.

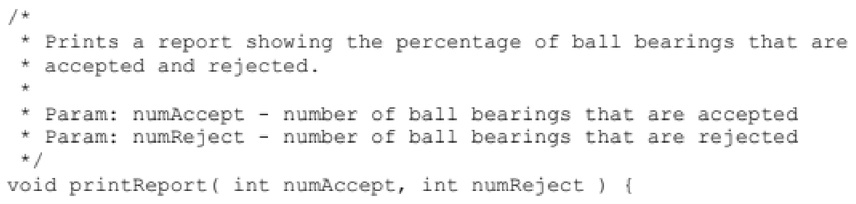
Carefully read through all of the provided code, including comment statements, before writing any code!

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

#define SOURCE "ballBearings.dat"

#define MIN_DIAM 9.95
#define MAX_DIAM 10.02

void doQualityControl( FILE* );
void printReport( int, int );

int main( void ) {
    FILE* inFile;

    inFile = fopen( SOURCE, "r" );

    if( inFile == NULL ) {
        printf( "Error opening input file, program terminating.\n" );
    }
    else {
        doQualityControl( inFile );

        fclose( inFile );
    }

    system( "PAUSE" );
    return 0;
}
```

```
/*
 * Performs quality control on ball bearing data.
 *
 * (Hint: calls printReport to print final report on screen)
 *
 * Param: inFile - points to file containing ball bearing data
 */
void doQualityControl( FILE* inFile ) {
```

Solution:

```
/*
* Performs quality control on ball bearing data.
*
* (Hint: calls printReport to print final report on screen)
*
* Param: inFile - points to file containing ball bearing data
*/

void doQualityControl( FILE* inFile ){
```



```
/*
 * Prints a report showing the percentage of ball bearings that are
 * accepted and rejected.
 *
 * Param: numAccept - number of ball bearings that are accepted
 * Param: numReject - number of ball bearings that are rejected
 */
void printReport( int numAccept, int numReject ) {
```

```
    double diameter;
    double weight;
    int countAccept = 0;
    int countReject = 0;

    while( fscanf( inFile, "%lf %lf", &diameter, &weight ) == 2 ) {
        if( diameter > MIN_DIAM && diameter < MAX_DIAM )
            countAccept++;
        else
            countReject++;
    }

    printReport( countAccept, countReject );
}

/*
* Prints a report showing the percentage of ball bearings that are
* accepted and rejected.
*
* Param: numAccept - number of ball bearings that are accepted
* Param: numReject - number of ball bearings that are rejected
*/

void printReport( int numAccept, int numReject ){
    int total;
    double percentAccept;
    double percentReject;

    total = numAccept + numReject;
    percentAccept = (double) numAccept / total * 100;
    percentReject = (double) numReject / total * 100;

    printf( "Percent accepted: %5.1f\n", percentAccept );
    printf( "Percent rejected: %5.1f\n", percentReject );
}
```

20. A global variable Assume we have a text file that contains an unknown number of rows of data. On each row there are three integer values, each separated from the next by a space. Write a function that takes a pointer to the file as its only parameter. The function must determine whether or not the entries on every row of the file are in decreasing order. If all rows of data have entries in decreasing order, the function must return TRUE otherwise it must return FALSE. Do not process more rows of data than necessary. Assume that the file has already been opened for reading. Further assume that TRUE and FALSE are symbolic constants defined with appropriate values.

Sample file:
```
3 2 1
3 0 -2
5 3 3
8 4 0
```

Function produces `TRUE` as entries on every row are in decreasing order.

Sample file:
```
3 2 1
3 0 -2
5 3 4
8 4 0
```

Function produces `FALSE` as at least one row (the 3rd) does not have its entries in decreasing order.

```
/*
* Determines whether all the rows of a file have their entries
* in decreasing order.
*
* Param: inFile     pointer to file that contains the integer values,
* already opened for reading
* Returns: TRUE if the data on every row of the file is in
* decreasing order, FALSE otherwise.
*/

int allRowsDecreasing( FILE* inFile ) {
```

Solution:

```
int numRowsDecreasing(FILE* inFile){
    int allDecreasing = TRUE;
    int num1, num2, num3;

    while(fscanf( inFile, "%d %d %d", &num1, &num2, &num3 ) == 3 && allDecreasing){
        if(num1 < num2 || num2 < num3)
        allDecreasing = FALSE;
    }

    return allDecreasing;
}
```

***NOTE: no writing to a file is included. Please find questions from class and attempt them.*

7 Arrays

21. Write a function countZerosInCols that takes a two dimensional array of integers and the number of rows in that array as parameters. The function takes a third parameter zerosInCol that is a one dimensional array whose size matches the number of columns in the two dimensional array. After a call to this function is made, zerosInCol[i] contains the number of entries in the ith column of the two dimensional array that are zero. We assume that NROWS and NCOLS are symbolic constants that have been defined to be the number of rows and number of columns in the two dimensional array, respectively.

    So, for example, suppose we declare the following arrays: int data[ ][NCOLS] = { { 2, 4, 5, 1 }, { 3, 0,

    2, 1 }, { 2, 4, 5, 1 }, { 3, 0, 2, 7 }, { 4, 3, 0, 6 } };
    int zerosInCol[NCOLS];

    and then make the function call:

    countZerosInCols(data, NROWS, zerosInCol);

    After this call ends, zerosInCol will have the values 0, 2, 1, 0 . Note that there are no zeros in the first column of data, 2 zeros in the second column, 1 in the third, etc.

    Note: in the interests of time, it is not necessary to write any comment statements.

Solution:

```
/*
* Param: data      2D array of data
* Param: nRows      number of rows of data
* Param: zerosInCol      array that will contain the number of zeros
* in each column of data by end of function call
*/

void countZerosInCols(int data[ ][NCOLS], int nRows,
int zerosInCol[ ]){
    int row;
    int col;
    int count;

    for(col = 0; col < NCOLS; col++){
        count = 0;

        for(row = 0; row < nRows; row++){
            if( data[row][col] == 0 )
                count++;
            }
            zerosInCol[ col ] = count;
        }
    }
```

22. Write a function that takes two input parameters. The first input parameter is a string (a char array), and the second parameter is a single character (char). The function counts how many times the given character appears in the string and returns this result. Your function must **NOT** use any of the functions in the string library (e.g. strlen), or any other functions.

Solution:

```
/*
* Param: array - array of character
* Param: searchFor - the character to search for
* Returns: number of times the character searchFor is found in array
*/

int search_and_count(char array[], char searchFor){
    int i = 0;
    int count = 0;

    while (array[i] != '\0'){
        if (array[i] == searchFor)
            count++;
            i++;
        }

    return count;
}
```

23. In this question you will develop a program that generates elementary math problems for kids. The program creates random problems using one of the operators '+', '-' and '*'. The operands (the numbers to operate on) are integers between 0 and 9 (including 0 and 9). Ten questions are generated and the user gets a maximum of 3 attempts at each question. The percentage of questions answered correctly is reported to the user before the program quits.

Here is part of a sample run showing interaction with the user for questions 2 and 3 only, as well as the final report printed after all questions have been attempted (recall that a total of 10 questions is asked each time the program is run). The reported score in this example assumes that the user came up with the right answer, within the maximum number of attempts, on 9 of the 10 questions asked. Study this output carefully.

```
<<Question #1 omitted>>

Question #2:
4 - 3 = ?

You have 3 attempts remaining.  Enter your answer...
2
Your answer is not correct.
You have 2 attempts remaining.  Enter your answer...
1
Correct!

Question #3:
7 * 4 = ?

You have 3 attempts remaining.  Enter your answer...
24
Your answer is not correct.
You have 2 attempts remaining.  Enter your answer...
7
Your answer is not correct.
You have 1 attempts remaining.  Enter your answer...
4
You have no more attempts remaining.
The correct answer is 28

<<Question #4 -> #10 omitted>>

Your score is 90.0 percent.
```

Your task is to complete the program presented on the following pages. To do this you must:

- insert code into the two underlined blank spaces provided in main

- complete the function generateQuestion by inserting code to determine the correct answer to the question and return it

- complete the function getUserAnswer

- complete the function processUserAnswers

It is recommended that you carefully read the documentation for all functions before you attempt to write any code.

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

#define NUM_OPERATORS 3    /* number of operators to choose from */
#define NUM_QUESTIONS 10   /* number of questions in quiz */
#define MAX_ATTEMPTS 3     /* max number of attempts to answer a question */
#define MAX_NUM 10         /* operands are in range 0 to (MAX_NUM - 1) */

#define FALSE 0
#define TRUE 1

int generateQuestion( int );
int processUserAnswers( int );
int getUserAnswer( int );
int getRandomOperand( void );

int main( void ) {
    int qNum;
    int correctAnswer;
    int numCorrect = 0;

    srand( (unsigned) time( NULL ) );

    for( qNum = 1; qNum <= NUM_QUESTIONS; qNum++ ) {
        //generate and display question

        correctAnswer = _____________________________________

        // Respond to user getting answer right or running out of attempts

        if( ________________________________________________ ) {
            printf( "Correct!\n\n" );
            numCorrect++;
        }
        else {
            printf( "You have no more attempts remaining.\n" );
            printf( "The correct answer is %d\n\n", correctAnswer );
        }

    }

    printf( "Your score is %.1f percent.\n",
                   100.0 * numCorrect / NUM_QUESTIONS );

    return 0;
}


/*
 *  Generates a random operand between 0 (inclusive) and MAX_NUM (exclusive)
 */
int getRandomOperand( void ) {
    return rand() % MAX_NUM;
}
```

```
/*
 *  Generates a new random question, prints it on the screen and returns
 *  correct answer.
 *
 *  Param:  qNum - the question number
 *  Return: answer to question
 */
int generateQuestion( int qNum ) {
    char operators[] = { '+', '-', '*' };
    char operator = operators[ rand() % NUM_OPERATORS ];
    int operand1 = getRandomOperand();
    int operand2 = getRandomOperand();
    int answer;

    printf( "Question #%d: \n", qNum );
    printf( "%d %c %d = ?\n\n", operand1, operator, operand2 );

    // determine correct answer to question just printed and return it










}

/*
 *  Tells the user how many attempts they have remaining and prompts them
 *  to enter their answer to the question.
 *
 *  Param:   attemptsRemaining - number of attempts remaining to answer
 *           the question
 *  Returns: user's answer to question
 */
int getUserAnswer( int attemptsRemaining ) {
```

```
/*
 *  Give user MAX_ATTEMPTS to enter correct answer. Tell user if
 *  answer is not correct and, if user has another attempt remaining,
 *  prompt for next answer.
 *
 *  Param:  correctAnswer - the correct answer to the question
 *  Return: TRUE if user enters correct answer within max number of attempts,
 *          FALSE otherwise.
 */
int processUserAnswers( int correctAnswer ) {
```

Solution:

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

#define NUM_OPERATORS 3 /* number of operators to choose from */
#define NUM_QUESTIONS 10 /* number of questions in quiz */
#define MAX_ATTEMPTS 3 /* maximum number of attempts to
answer a question */
#define MAX_NUM 10 /* operands are in range 0 to (MAX_NUM - 1) */

#define FALSE 0
#define TRUE 1

int generateQuestion( int );
int processUserAnswers( int );
int getUserAnswer( int );
int getRandomOperand( void );

int main( void ) {
    int qNum;
    int correctAnswer;
    int numCorrect = 0;

    srand( (unsigned) time(NULL) );

    for( qNum = 1; qNum <= NUM_QUESTIONS; qNum++ ){
        //generate and display question
        correctAnswer = generateQuestion( qNum );

        // Respond to user getting answer right or running out of attempts
        if( processUserAnswers( correctAnswer ) ) {
            printf( "Correct!\n\n" );
            numCorrect++;
        }
        else{
            printf( "You have no more attempts remaining.\n" );
            printf( "The correct answer is %d\n\n", correctAnswer );
        }
    }

    printf( "Your score is %.1f percent.\n", 100.0 * numCorrect / NUM_QUESTIONS );

    return 0;
}
```

```c
/*
* Generates a new question and prints it on the screen.
*
* Param: qNum - the question number
* Return: answer to question
*/

int generateQuestion( int qNum ){
    char operators[ ] = { '+', '-', '*' };
    char operator = operators[ rand() % NUM_OPERATORS ];
    int operand1 = getRandomOperand();
    int operand2 = getRandomOperand();
    int answer;

    printf( "Question #%d: \n", qNum );
    printf( "%d %c %d = ? \n \n", operand1 , operator , operand2 );

    if( operator == '+' )
        answer = operand1 + operand2;
    else if( operator == '-' )
        answer = operand1 - operand2;
    else
        answer = operand1 * operand2;

    return answer;
}
```

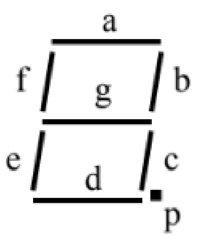
```
/*
* Generates a random operand between 0 (inclusive) and MAX_NUM (7-selusive)
*/

int getRandomOperand( void ){
    return rand() % MAX_NUM;
}
```

```
/*
* Give user MAX_ATTEMPTS to enter correct answer.
*
* Param: correctAnswer - the correct answer to the question
* Return: TRUE if user enters correct answer within maximum number of
* attempts, FALSE otherwise.
*/

int processUserAnswers( int correctAnswer ){
    int userAnswer;
    int attemptNum = 0;

    // give user maximum attempts to answer question
    userAnswer = getUserAnswer( MAX_ATTEMPTS );
    attemptNum++;

    while( userAnswer != correctAnswer && attemptNum < MAX_ATTEMPTS){
        printf( "Your answer is not correct.\n" );
        userAnswer = getUserAnswer( MAX_ATTEMPTS - attemptNum);
        attemptNum++;
    }

    return userAnswer == correctAnswer;
}
```

```
/*
* Tells the user how many attempts they have remaining and prompts them
* to enter their answer to the question.
*
* Param: attemptsRemaining - number of attempts remaining to answer
* the question
* Returns: user's answer to question
*/

int getUserAnswer( int attemptsRemaining ){
    int answer;

    printf( "You have %d attempts remaining. Enter your answer...\n",
    attemptsRemaining );
    scanf( "\%d", \&answer );

    return answer;
}
```

8 Seven Segment Displays

24. The following diagrams associated with 7-segment displays will be useful for the questions below. Recall that a 0 turns a segment off, and a 1 turns a segment on.



a	b	c	d	e	f	g	p

(a) Determine the data value to write to the 7-segment display to show the uppercase letter ”U”, and express this value both as a binary number and a decimal number.

(b) If the value 224 is written to the 7-segment display, what is the digit from 0 to 9 (including 0 and 9) that will be displayed on the 7-segment display? For full marks, you must show your work.

Solution:

(a) In binary: 01111100

In decimal: 64+32+16+8+4 = 255-127-2-1=124

Note that the question specifically asks for uppercase and not lowercase, so segments f and b are also turned on.

(b) 224 in decimal is 11100000 in binary. To get full marks, you must show the steps, i.e.,

224%2=0, 224/2=112

112%2=0, 112/2=56

etc.

or otherwise explain how you got the binary number By using the provided figures, 11100000 means segments a, b and c are on and the rest of segments are all off, which gives the digit 7.

25. *(Note: you may not have done this in class but it is good practice anyway)*

The writeNumber function that was developed in class did not handle negative values. Modify the function to be able to correctly display negative values. Assume the same LED screen that was used in class (i.e. eight 7-segment displays arranged in a row, with the right-most LED display at position 0, and the left-most LED display at position 7).

The new version of the function will display at most the lower 7 digits of the given value. The negative sign (if the number is negative) will be displayed on the LED display immediately to the left of the number that is displayed. If the number is positive, that LED display will be left blank (i.e. a space is written).

For the purpose of this question, assume that the integer divide (/) and modulus (%) operators only work correctly on positive integer values. They do not necessarily work as expected when the operands are negative integer values.

(a) Determine the data value to write to the 7-segment display to show the negative sign, and express this value both as a binary number and a decimal number. You may find it useful to refer to diagrams in the previous question.

(b) The original version of the writeNumber function is given to you on the next page. You do not have to re-write the complete function. It is sufficient to edit the given function:

- to indicate which lines are deleted, circle the line(s) and write delete beside them;
- the indicate which lines are added, write the new lines in the space provided to the right of the code listing, and indicate with an arrow where you would put the new lines in the function;
- to indicate a line that is changed, delete the line and add a new line using the above procedures.

You can assume that the writeDigit function is already implemented, and that it will use the displayWrite function to display the given digit at the given position on the LED screen. The valid digits are all the digits from 0 to 9, including 0 and 9.

```c
/* Number of displays in the LED screen. */
#define NUM_DISPLAYS 8

/* A SPACE character turns all segments off */
#define TURNOFF 0

void writeNumber(int number)
{
    int pos;
    int digit;

    /* start at the right */
    pos = 0;

    /*
     * Extract one digit at a time, until we have no more
     * digits left (i.e. number==0) or we have exhausted our
     * LED displays (i.e. pos < NUM_DISPLAYS).
    */
    do {
        digit = number % 10;
        number = number / 10;

        writeDigit(digit, pos);

        /* determine the position for the next digit, if any */
        pos++;
    } while ( pos < NUM_DISPLAYS && number != 0 );


    /*
     * If there are any LED displays left, then turn them off
    */
    while (pos < NUM_DISPLAYS)
    {
        displayWrite(TURNOFF, pos);
        pos++;
    }
}
```

Solution:

The function is given to you. As per instructions, you only need to modify (add, delete or change) the statements.

```
/* Number of displays in the LED screen. */
#define NUM_DISPLAYS 8

/* A SPACE character turns all segments off */
#define TURNOFF 0

/* Add this definition */
#define MINUS_SIGN 2

void writeNumber(int number)
{
    int pos;
    int digit;

    /* Add the following code */
    int negNumber;

    if ( number < 0 )
    {
        negNumber = TRUE;
        number = -number;
    }
    else
        negNumber = FALSE;
    /*************************/
    /* start at the right */
    pos = 0;

    do {
        digit = number % 10;
        number = number / 10;

        writeDigit(digit, pos);
        pos++;

    /* Change the following one line from
     * } while ( pos < NUM_DISPLAYS && number != 0 );
     * to:
    */
    } while ( pos < NUM_DISPLAYS-1 && number != 0 );

    /* Add the following if */
    if ( negNumber )
    {

        displayWrite(MINUS_SIGN, pos);
        pos++;
    }
```

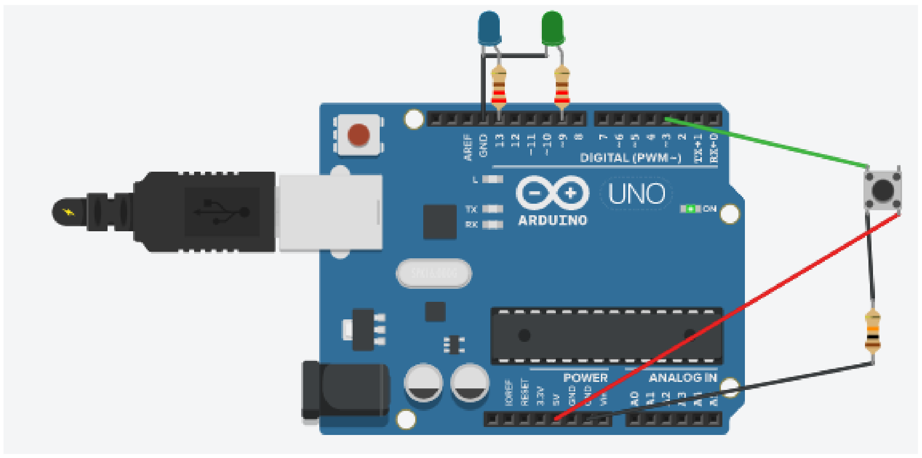
9 Arduino

Use TinkerCad for the code / simulations

Note that TinkerCad does not really care about bracket placement, so do not panic if the solutions seem to have weird bracket orientation.

26. Create a system of two LEDs and a button to control them. Pushing the button once should light up LED #1 once. Pushing the button again should light up LED #2 twice. Pushing the button once more should light up LED #1 three times....etc

Solution: Your setup should look something like this:

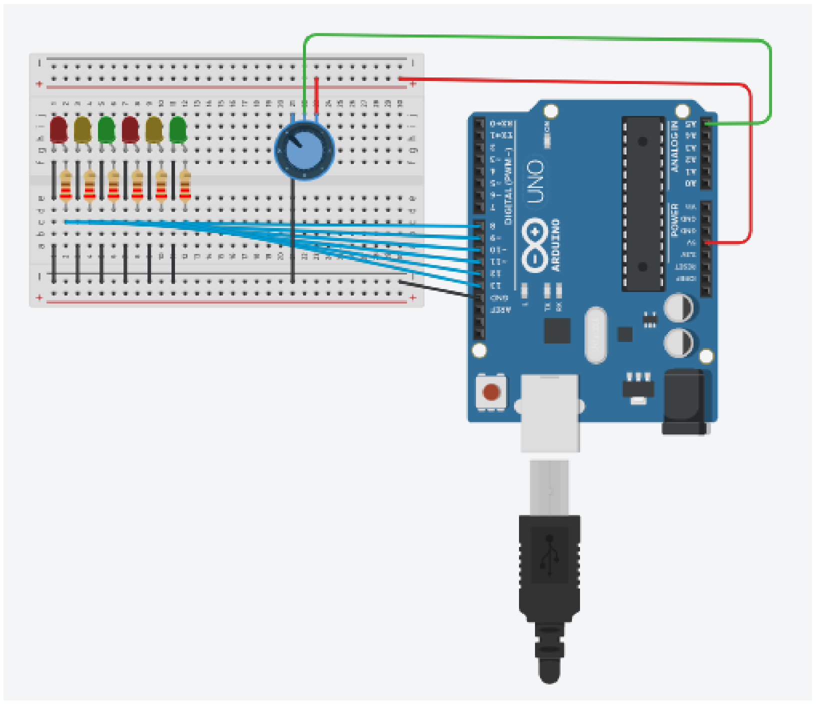


Solution:

```
#define BLUE_LED 13
#define GREEN_LED 9
#define BUTTON 3

void blinkBlue (int count){
    for (int i=0; i<count ;i++){
        digitalWrite(BLUE_LED,HIGH);
        delay(500);

        digitalWrite(BLUE_LED,LOW);
        delay(500);
    }
}

void blinkGreen (int count){
    for (int i=0; i<count; i++){
        digitalWrite(GREEN_LED,HIGH);
        delay(500);

        digitalWrite(GREEN_LED,LOW);
        delay(500);
    }
}

void setup()
{
    pinMode(BLUE_LED, OUTPUT);
    pinMode(GREEN_LED, OUTPUT);
    pinMode(BUTTON, INPUT);
}

int count=1;

void loop(){
    int buttonState=digitalRead(BUTTON);

    if(buttonState==HIGH){
        if(count%2==0){
            blinkGreen(count);
        }
        else if (count%2!=0){
            blinkBlue(count);
        }
    count++;
    }
}
```

27. Create a system of 6 LEDs in a line. Code a system to make the lights flash one by one in a line. Use a Potentiometer to control the speed of the lights flashing.

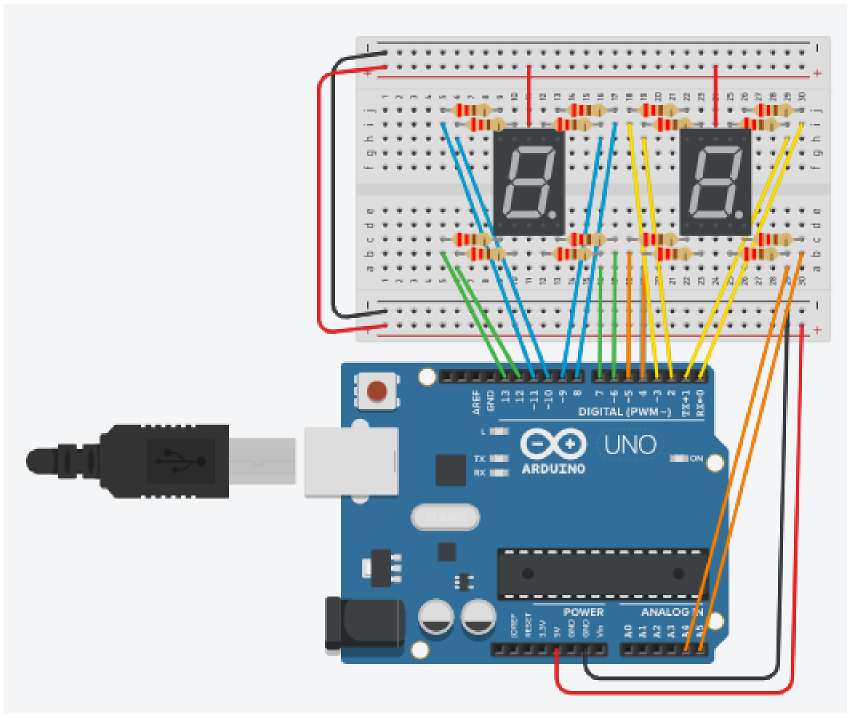
Solution: Your setup should look something like this:



```
#define num_LEDs 6
#define POTENTIO A5

int LED[]={8,9,10,11,12,13};

void setup(){

for(int i=0; i<num_LEDs; i++){
    pinMode(LED[i], OUTPUT);
    {
    pinMode(POTENTIO, INPUT);
    Serial.begin(9600);
}

int count=0;

void loop(){
    int sensor=analogRead(POTENTIO);
    //Serial.println(sensor);

    float hold = sensor/1023.*900+100;
    Serial.println(hold);

    if (count%2==0){
        for (int i=0; i<(num_LEDs);i++){
            digitalWrite(LED[i],HIGH);
            delay(hold);
            digitalWrite(LED[i],LOW);
            delay(hold);
        }
        count++;
    }
    else{
        for (int i=(num_LEDs-2); i>0;i--){
            digitalWrite(LED[i],HIGH);
```

```
                delay(hold);
                digitalWrite(LED[i],LOW);
                delay(hold);
            }
            count++;
        }
    }
```

28. Setup this Arduino system:



Make this system spell out APSC 160 on the first 7 segment display.

Then, using both displays, display your favourite 2 digit number.

Solution:

```
#define A1 9
#define B1 8
#define C1 7
#define D1 12
#define E1 13
#define F1 10
#define G1 11
#define DP1 6
#define A2 1
#define B2 0
#define C2 A4
#define D2 4
#define E2 5
#define F2 2
#define G2 3
```

```
#define DP2 A5
#define NUM_LEDS 7

const int led1[ ]={A1,B1,C1,D1,E1,F1,G1,DP1};
const int led2[ ]={A2,B2,C2,D2,E2,F2,G2,DP2};

// Which LEDs should be on/off for specific letters
const int num[7][8]=
{{0,0,0,1,0,0,0,1},  // A
 {0,0,1,1,0,0,0,1},  // P
 {0,1,0,0,1,0,0,1},  // S
 {0,1,1,0,0,0,1,1},  // C
 {1,0,0,1,1,1,1,1},  // 1
 {0,1,0,0,0,0,0,1},  // 6
 {0,0,0,0,0,0,1,1}}; // 0

// function for first LED display
void lightItUp1(const int ledOnOff1[ ]);

// function for second LED display
void lightItUp2(const int ledOnOff1[ ]);

// Setup LEDs for both displays
void setup(){
    for (int i=0;i<NUM_LEDS;i++){
        pinMode(led1[i],OUTPUT);
   }
    for (int i=0;i<NUM_LEDS;i++){
        pinMode(led2[i],OUTPUT);
    }
}
```

Solution: ....continued

```
void loop(){
    // Display APSC160 on left display
    for (int i=0;i<7;i++){ //the 7 here is for the chars in apsc160
        lightItUp1(num[i]);
        delay(1000);
    }
    delay(40);

    //Display 1 on left display and 0 on right
    //display for long period of time.
    lightItUp1(num[4]);
    delay(10);
    lightItUp2(num[6]);
    delay(5000);
}

// write APSC160,1 on first display
void lightItUp1(const int ledOnOff1[]){
    for (int i=0;i<NUM_LEDS;i++){
        digitalWrite(led1[i],ledOnOff1[i]);
    }
}

//write 0 on second display
void lightItUp2(const int ledOnOff1[]){
    for (int i=0;i<NUM_LEDS;i++){
        digitalWrite(led2[i],ledOnOff1[i]);
    }
}
```