Introduction:

The creation of advanced computation and programming has provided easier solutions to solving real-life problems through programs. Programming has been used in many professional sectors such as healthcare, engineering, business, and manufacturing. Throughout this past semester, the basics of the C programming language have been taught and their applications have been reviewed. In the following report, three real-life examples will be studied and programs will be created for them. The first example was done by Kevin and Julibeb. The goal of the first example was to create a C Program with the switch function included in the main function, as this would allow several different options for calculating several different intravenous rates. The second example is a binary search algorithm. The objective of the program is to search for a value inserted by the user by checking the middle value of an array of numbers. If the number inserted is not found, the program checks if the number inserted is greater or less than the middle number. It reindexes and repeats checking the middle value and reindexing until it finds the inserted number or returns an error message stating the inserted number was not found. The second program was made by Janice and Joao Paulo. The third and final program was completed by Josstina and Joao Paulo. The program allows a user to insert the three colors of bands used for resistors. The program identifies the values of each resistor band in Ohms and converts them to kilo-Ohms. Below are the codes and outputs for each of the described examples.

Question 1:
Code of Program:

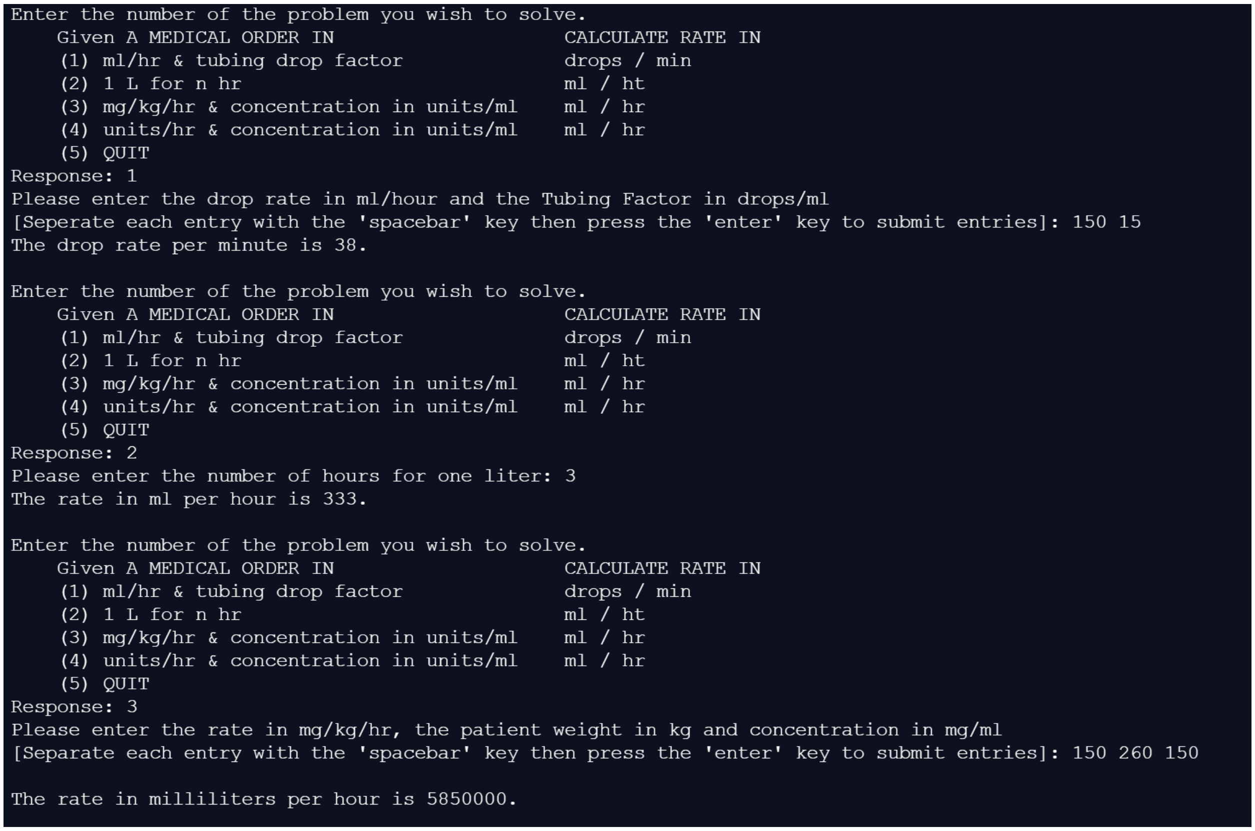
```
#include <stdio.h>
#define MINUTE 60 /*Number of minutes in an hour */
#define M_Liter 1000
#define SENTINEL 5


int get_problem(void);
double get_n_nhours(double num_hours); /*To be completed */
void get_rate_drop_factor(double * /*pointers */, double * /*pointers */);
void get_kg_rate_conc(double *, double *, double *);
void get_units_conc(double *, double *);
double fig_drops_min(double ,double );
double fig_ml_hr(double );
double by_weight(double,double, double);
double by_units(double, double);

int main(void) {
 int value;
 double answer;
 double ml_hour;
 double drops_ml;
 double mg_kg_hour;
 double pat_weight;
 double mg_ml;
 double units_hour;
 double units_ml;
 double num_hours;
 value = get_problem();
 while(value !=SENTINEL){
   switch (value){
     case 1:
     /* ml/hr & tubing drop factor 8 */
     /* drops/min */
     fig_drops_min(ml_hour,drops_ml);
     printf("\n");
     break;
     case 2:
```

```
     /* 1 L for n hr */
     /* ml/hr */
     fig_ml_hr(num_hours);
     printf("\n");

     break;
     case 3:
     by_weight(mg_kg_hour, pat_weight, mg_ml);
     printf("\n");
     break;
     case 4:
     by_units(units_hour, units_ml);
     printf("\n");
     break;
     case 5:
     printf("Program will be terminated");
     break;
     /* uses the variables declared within the main() program and the
given function prototypes,complete the body of the switch statement */
     default:
     printf("Wrong input. \n");
   }
   value = get_problem();
 }
 return 0;
}

/* function that displays menu and gets user's input */
int get_problem(void){
 int menu_number;
  printf("Enter the number of the problem you wish to solve.\n");
 printf("\tGiven A MEDICAL ORDER IN\t\t\t\t\tCALCULATE RATE IN\n");
 printf("\t(1) ml/hr & tubing drop factor\t\t\t\tdrops / min\n");
 printf("\t(2) 1 L for n hr\t\t\t\t\t\t\tml / ht\n");
 printf("\t(3) mg/kg/hr & concentration in units/ml\tml / hr\n");
 printf("\t(4) units/hr & concentration in units/ml\tml / hr\n");
 printf("\t(5) QUIT\n");
 printf("Response: ");
 scanf("%d", &menu_number);
 return (menu_number);
```

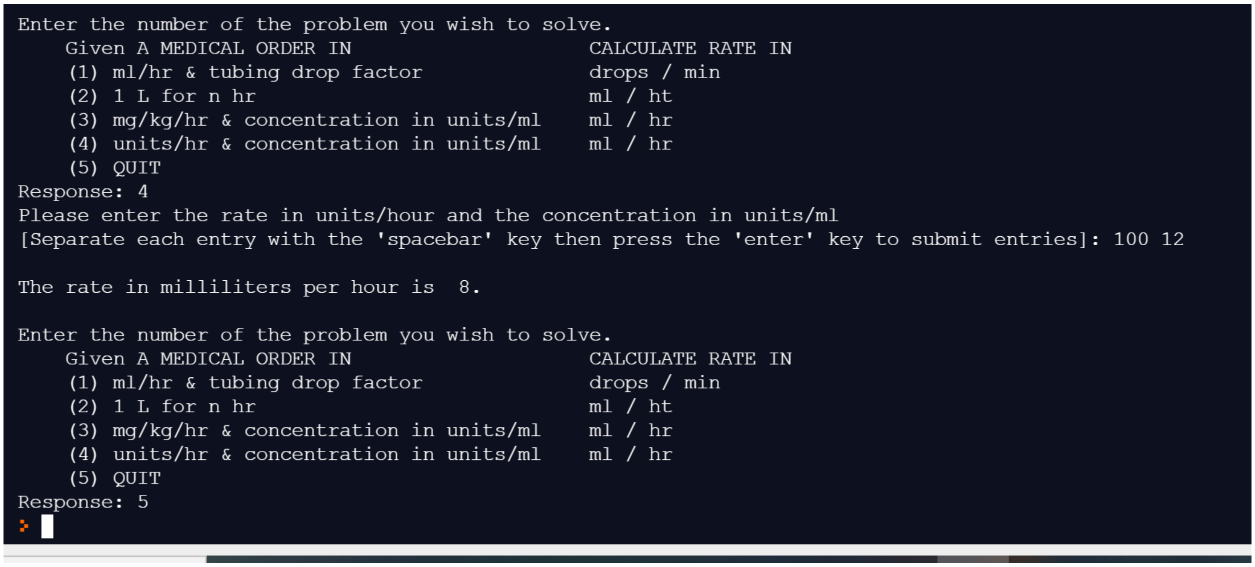
```
}

/*function to get the number of hours */
double get_n_hours(double num_hours){
 printf("Please enter the number of hours for one liter: ");
 scanf("%lf", &num_hours);
 return num_hours;
}

/* function prompts the user to enter rate and tubing's drop factor then
returns values through output parameters */
void get_rate_drop_factor(double *ml_hour, double *drops_ml){
 printf("Please enter the drop rate in ml/hour and the Tubing Factor in
drops/ml \n[Seperate each entry with the 'spacebar' key then press the
'enter' key to submit entries]: ");
 scanf("%lf %lf",ml_hour, drops_ml);
}

/* function prompts for rate, patients weight, and concentration then
returns values through output parameters */
void get_kg_rate_conc(double *mg_kg_hour, double *pat_weight, double
*mg_ml){
 printf("Please enter the rate in mg/kg/hr, the patient weight in kg and
concentration in mg/ml \n[Separate each entry with the 'spacebar' key then
press the 'enter' key to submit entries]: ");
 scanf("%lf %lf %lf", mg_kg_hour, pat_weight, mg_ml);
}

/* function prompts for rate and concentration then returns values through
output parameters */
void get_units_conc(double *units_hour, double *units_ml){
 printf("Please enter the rate in units/hour and the concentration in
units/ml \n[Separate each entry with the 'spacebar' key then press the
'enter' key to submit entries]: ");
 scanf("%lf %lf", units_hour, units_ml);
}

/* function takes as input rate and concentration then returns as its
value the result of dividing their product by MINUTE */
double fig_drops_min(double ml_hour, double drops_ml){
```

```c
 get_rate_drop_factor(&ml_hour, &drops_ml);
 double answer = (ml_hour*drops_ml)/MINUTE;
 printf("The drop rate per minute is %2.lf. \n", answer);
 return answer;
}

/* function takes as input num_hours and returns as its value the quotient
of *100 and num_hours */
double fig_ml_hr(double num_hours){
 double answer = M_Liter/get_n_hours(num_hours);
 printf("The rate in ml per hour is %2.lf. \n", answer);
 return answer;
}

/* function takes 3 inputs and returns as its value the product of rate
and patient's weight divided by concentration */
double by_weight(double mg_kg_hr, double pat_weight, double mg_ml){
 get_kg_rate_conc(&mg_kg_hr, &pat_weight, &mg_ml);
 double answer = (mg_kg_hr*pat_weight*mg_ml);
 printf("\nThe rate in milliliters per hour is %2.lf. \n",answer);
 return answer;
}

/* function takes 2 inputs and returns as its value the quotient of
units_hr and units_ml*/
double by_units(double units_hour, double units_ml){
 get_units_conc(&units_hour,  &units_ml);
 double answer = (units_hour/units_ml);
 printf("\nThe rate in milliliters per hour is %2.lf. \n",answer);
 return answer;
}
```

Output:



```
Enter the number of the problem you wish to solve.
    Given A MEDICAL ORDER IN                        CALCULATE RATE IN
    (1) ml/hr & tubing drop factor                  drops / min
    (2) 1 L for n hr                                ml / ht
    (3) mg/kg/hr & concentration in units/ml        ml / hr
    (4) units/hr & concentration in units/ml        ml / hr
    (5) QUIT
Response: 1
Please enter the drop rate in ml/hour and the Tubing Factor in drops/ml
[Seperate each entry with the 'spacebar' key then press the 'enter' key to submit entries]: 150 15
The drop rate per minute is 38.

Enter the number of the problem you wish to solve.
    Given A MEDICAL ORDER IN                        CALCULATE RATE IN
    (1) ml/hr & tubing drop factor                  drops / min
    (2) 1 L for n hr                                ml / ht
    (3) mg/kg/hr & concentration in units/ml        ml / hr
    (4) units/hr & concentration in units/ml        ml / hr
    (5) QUIT
Response: 2
Please enter the number of hours for one liter: 3
The rate in ml per hour is 333.

Enter the number of the problem you wish to solve.
    Given A MEDICAL ORDER IN                        CALCULATE RATE IN
    (1) ml/hr & tubing drop factor                  drops / min
    (2) 1 L for n hr                                ml / ht
    (3) mg/kg/hr & concentration in units/ml        ml / hr
    (4) units/hr & concentration in units/ml        ml / hr
    (5) QUIT
Response: 3
Please enter the rate in mg/kg/hr, the patient weight in kg and concentration in mg/ml
[Separate each entry with the 'spacebar' key then press the 'enter' key to submit entries]: 150 260 150

The rate in milliliters per hour is 5850000.
```

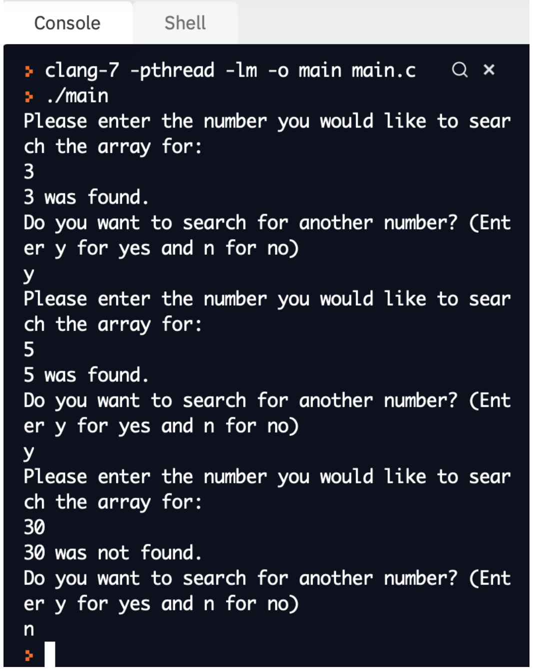


```
Enter the number of the problem you wish to solve.
    Given A MEDICAL ORDER IN                        CALCULATE RATE IN
    (1) ml/hr & tubing drop factor                  drops / min
    (2) 1 L for n hr                                ml / ht
    (3) mg/kg/hr & concentration in units/ml        ml / hr
    (4) units/hr & concentration in units/ml        ml / hr
    (5) QUIT
Response: 4
Please enter the rate in units/hour and the concentration in units/ml
[Separate each entry with the 'spacebar' key then press the 'enter' key to submit entries]: 100 12

The rate in milliliters per hour is  8.

Enter the number of the problem you wish to solve.
    Given A MEDICAL ORDER IN                        CALCULATE RATE IN
    (1) ml/hr & tubing drop factor                  drops / min
    (2) 1 L for n hr                                ml / ht
    (3) mg/kg/hr & concentration in units/ml        ml / hr
    (4) units/hr & concentration in units/ml        ml / hr
    (5) QUIT
Response: 5
```

Question 2:
Code of Program:

```
#include <stdio.h>
#define SIZE_OF_ARRAY 10
#define TRUE 1
#define FALSE 0
#define NOT_FOUND -1

int binary_srch(const int search_array[], int target, int size);

int main(void) {
  int target, index, num[10] = {1,3,5,6,12,63,78,95,101,130};
 char ans = 'y';

 do{

 printf("Please enter the number you would like to search the array for: 
\n");
 scanf("%d", &target);
  index = binary_srch(num, target, SIZE_OF_ARRAY);

 if(index == TRUE){
   printf("%d was found. \n", target);
 }

 else{
   printf("%d was not found. \n", target);
 }
  printf("Do you want to search for another number? (Enter y for yes and n 
for no) \n");
 scanf(" %c", &ans);
  }while(ans == 'y');
  return 0;
}

int binary_srch(const int search_array[], int target, int size){
  int top, bottom, middle, found;
 bottom = 0;
 top = size - 1;
 found = FALSE;

  middle = (top+bottom)/2;

  while(bottom <= top)
   {
       if(search_array[middle] < target)
           bottom = middle+1;

       else if(search_array[middle]==target)
       {
      found = TRUE;
           return found;
       }
       else
           top = middle-1;
       middle = (bottom+top)/2;
   }

found = NOT_FOUND;
return found;
 }
```

Output:

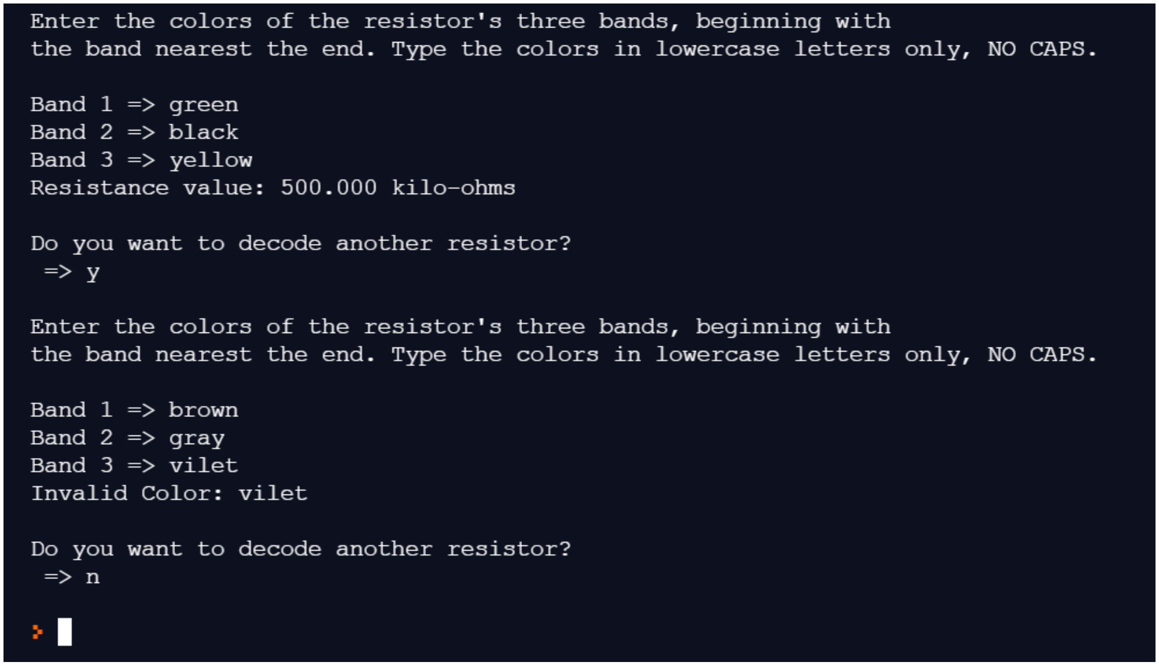


```
> clang-7 -pthread -lm -o main main.c
> ./main
Please enter the number you would like to sear
ch the array for:
3
3 was found.
Do you want to search for another number? (Ent
er y for yes and n for no)
y
Please enter the number you would like to sear
ch the array for:
5
5 was found.
Do you want to search for another number? (Ent
er y for yes and n for no)
y
Please enter the number you would like to sear
ch the array for:
30
30 was not found.
Do you want to search for another number? (Ent
er y for yes and n for no)
n
>
```

Question 3:
Code of Program:

```
#include <stdio.h>
#include <math.h> /* for pow*/
#include <ctype.h> /* for toupper*/
#include <string.h> /* for strlen*/
#define NOT_FOUND -1 /* constants */
#define SUB_1 10
#define SUB_2 7

int search(const char [][SUB_2], const char [], int);

int main(void){
 char reply, /* user reply*/
 char_left; /* character left in the input stream*/
 int i;
 int counter; /* counters */
 int value; /* subscript of target found in list*/
 double answer = 0.0; /* value of resistor in kilo-ohms*/
 int no_error = 1; /* denotes no error */

/* initializing the array*/
char COLOR_CODES[SUB_1][SUB_2] = {"black", "brown","red",
"orange","yellow", "green","blue", "violet", "gray", "white"}; char
target[SUB_2]; /* target string array*/

do{
 no_error=1;
 printf("Enter the colors of the resistor's three bands, beginning
with\n");
 printf("the band nearest the end. Type the colors in lowercase letters
only, ");
 printf("NO CAPS.\n\n");

 for(counter = 1; counter<=3 ; counter++) {
     printf("Band %d => ", counter);
     scanf("%s", target);

     value = search(COLOR_CODES, target, SUB_1);
     /* searches for string*/
     if(value != NOT_FOUND) {
       switch(counter){
         case 1:
         answer = value * 10;
         break;
         case 2:
         answer += value;
         break;
         case 3:
         if (value > 3)
          answer*=pow(10, value-3);
        else
          for(i = 0; i < 3 - value ; i++)
             answer /= 10;
              }
       }
else{
  no_error = 0; /* if string not found*/
  break;
}
}
if (no_error==1)
printf("Resistance value: %.3f kilo-ohms\n\n", answer);
else
printf("Invalid Color: %s\n\n", target);

printf("Do you want to decode another resistor?\n => ");
scanf("%c%c", &char_left, &reply);
printf("\n");
} while(reply=='y');
}

/* function takes as input a list of strings, its size, and a target
string. Then, searches the list for the
target and returns as its value the subscript of the target in the list.
It returns -1 if target is not found.
*/
int search(const char COLOR_CODES[][SUB_2], const char target[], int
size){
int i, j; /* counters */
int length, counter = 0;
int found = 0; /* indicates when string is found*/
int where = 0; /* location of target*/
length = strlen(target);
for(i=0; i < size && found != 1  ; i++) {
 for(j = 0; j < length; j++)
    if(COLOR_CODES[i][j] == target[j])
      counter++;
 if(counter == length)
    found = 1;
 else
    counter=0;
}
--i;
if(found)
 where = i;
else
 where = -1;
return where;
}
```

Output:



```
Enter the colors of the resistor's three bands, beginning with
the band nearest the end. Type the colors in lowercase letters only, NO CAPS.

Band 1 => green
Band 2 => black
Band 3 => yellow
Resistance value: 500.000 kilo-ohms

Do you want to decode another resistor?
 => y

Enter the colors of the resistor's three bands, beginning with
the band nearest the end. Type the colors in lowercase letters only, NO CAPS.

Band 1 => brown
Band 2 => gray
Band 3 => vilet
Invalid Color: vilet

Do you want to decode another resistor?
 => n
```

Conclusion:

To conclude, this report offered an in-depth understanding of how programming languages and computation can be applied in modern day situations. It helped us understand how important programming is in daily life and how much it has and is able to contribute on a daily basis. Although the above examples are relatively simple, it is understood that with an advanced knowledge of the C language, programs with more advanced features and implementations can be created for almost any possible real-life scenario. If our group were able to do one thing differently about this project, it would most likely be able to do the code for the programs from the beginning without a template. There were a few times the template was either wrong or the syntax used was confusing or over complicated for the operation that needed to be completed. Otherwise, our group enjoyed doing this project and we have developed a deeper understanding of the applications of programming.