Name: Angelique Margaret M. Ardena

CMSC 21 – Machine Problem 1

1. The code shows a sequence of numbers being multiplied by two (but only until 128). In the code, it shows a number being multiplied by two, then the product of the first number will be multiplied by two once more, and so on and so forth, giving out numbers that are powers of two.

   Code:

```c
#include <stdio.h>

int main(void)
{
 int i;
 i = 1;
 while (i <= 128) {
 printf("%d ", i);
 i *= 2;
 }
 return 0;
}
```

   Output:

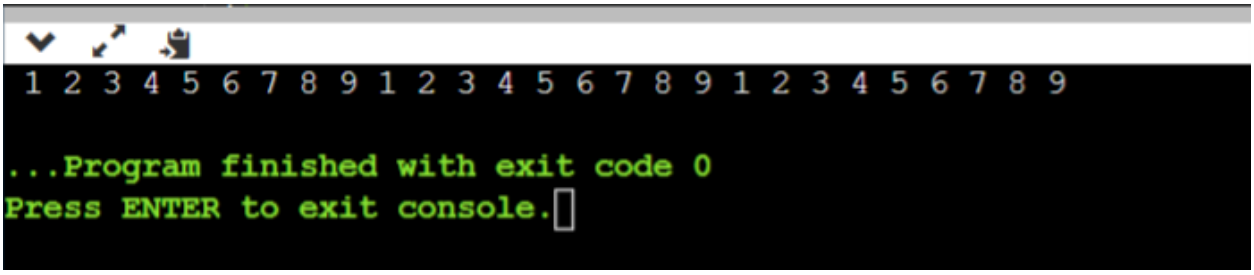
```
1 2 4 8 16 32 64 128

...Program finished with exit code 0
Press ENTER to exit console.
```

2. The do-while loop appears to have a different approach when compared to both for and while loops. A for loop and a while loop checks the condition first before the iteration. Meanwhile, a do-while loop executes a condition first before checking the condition if it still applies.

   Code:

```c
#include <stdio.h>

int main(void)
{

// do-while
 int i;
 i = 1;
 do{
     printf(" %d", i);
     i += 1;
 }
 while(i < 10);


//for loop
 int j;
 j = 1;
 for(; j< 10;){
     printf(" %d", j);
     j += 1;
 }

// while loop
 int k;
 k = 1;
 while(k < 10){
     printf(" %d", k);
     k += 1;
 }
 return 0;
}

```

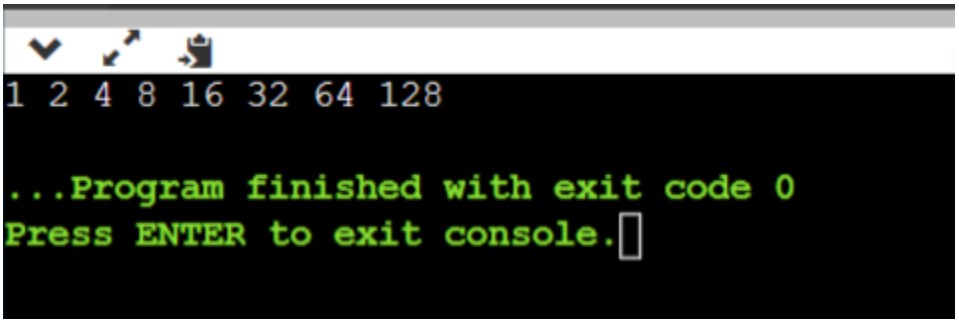
Output:



3. Code:

```c
#include <stdio.h>
int main(void)
{
 for(int i = 1; i <= 128; i *= 2) {
    printf("%d ", i);
 }
 return 0;
}
```

Output:



Comparison to #1:

```c
#include <stdio.h>
int main(void)
{
 for(int i = 1; i <= 128; i *= 2) {
    printf("%d ", i);
 }
 return 0;
}
```

```
1 2 4 8 16 32 64 128

...Program finished with exit code 0
Press ENTER to exit console.
```

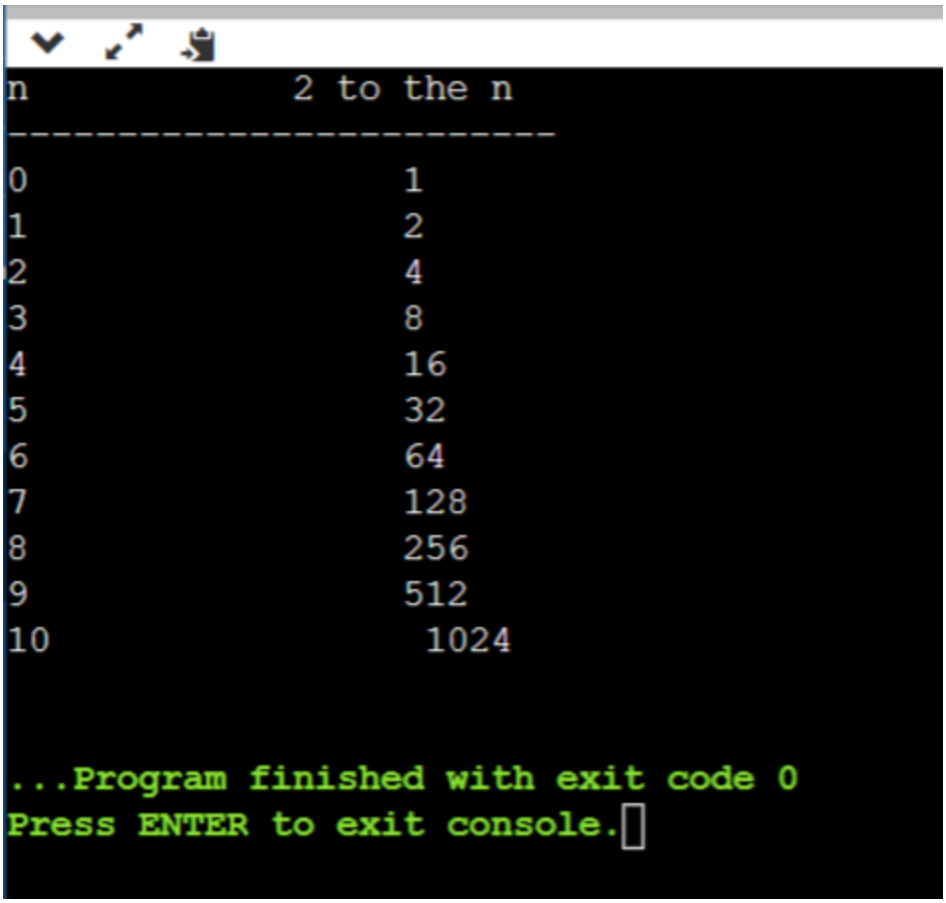
```c
#include <stdio.h>
int main(void)
{
 int i;
 i = 1;
 while (i <= 128) {
 printf("%d ", i);
 i *= 2;
 }
 return 0;
}
```

```
1 2 4 8 16 32 64 128

...Program finished with exit code 0
Press ENTER to exit console.
```

4. Code:

```c
#include<stdio.h>
int main()
{
    int b = 2, n = 0;
    long int p=1;

    printf("n            2 to the n\n");
    printf("-------------------------\n");

    while(n < 11)
    {
       if (n == 0)
           p = 1;

       else
        p *= b;
        printf("%d                %ld\n", n, p);
        ++n;
    }



    return 0;
}
```

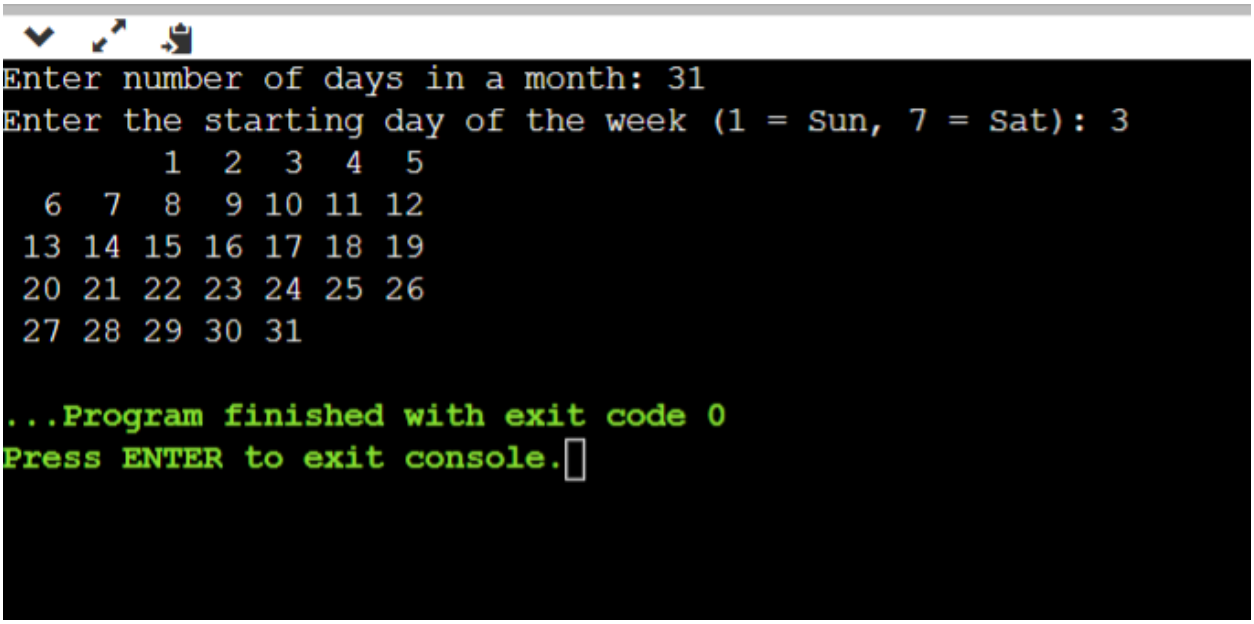
Output:


n 2 to the n
0 1
1 2
2 4
3 8
4 16
5 32
6 64
7 128
8 256
9 512
10 1024
...Program finished with exit code 0
Press ENTER to exit console.

5. Code:

```c
#include <stdio.h>
int main(void)
{
    int ndays, start, i;

    printf("Enter number of days in a month: ");
    scanf("%d", &ndays);

    if(ndays != 28 && ndays!= 29 && ndays != 30 && ndays != 31)
    printf("Invalid number. Type a valid number of days.");

    else
    {

    printf("Enter the starting day of the week (1 = Sun, 7 = Sat): ");
    scanf("%d", &start);

        if(start <= 0 || start >= 8)
        printf("Invalid number. Type a starting day.");

        else
        {
              for(i = 1; i < start; i++)
            printf("    ");

            for(i = 1; i <= ndays; i++)
            {
                printf("%3d", i);
                if((start + i - 1)%7 == 0)
                printf("\n");
            }
        }

    }

    return 0;
}
```

Output:



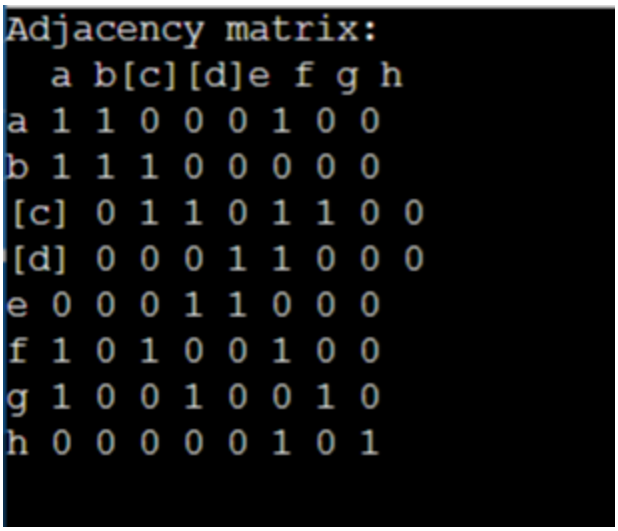
6. a. bool pathway[8] = {[0] = true, [2] = true};

   b. bool pathway[8] = {true, false, true};

7. Code:

```c
#include <stdio.h>

#define ROAD_POINTS 8

int main(){

    int road_networks[ROAD_POINTS][ROAD_POINTS] = {

     {1, 1, 0, 0, 0, 1, 0, 0},
     {1, 1, 1, 0, 0, 0, 0, 0},
     {0, 1, 1, 0, 1, 1, 0, 0},
     {0, 0, 0, 1, 1, 0, 0, 0},
     {0, 0, 0, 1, 1, 0, 0, 0},
     {1, 0, 1, 0, 0, 1, 0, 0},
     {1, 0, 0, 1, 0, 0, 1, 0},
     {0, 0, 0, 0, 0, 1, 0, 1}

    };

    //Displays adjacency matrix

    char letters[ROAD_POINTS] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'};

    printf("Adjacency matrix:\n");
    printf("  a b[c][d]e f g h\n");
    for (int i = 0; i < ROAD_POINTS; i++) {
        if (i == 2 || i == 3){
            printf("[%c] ", letters[i]);
        }else{
            printf("%c ", letters[i]);
        }
        for (int j = 0; j < ROAD_POINTS; j++) {
            printf("%d ", road_networks[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Output:


Adjacency matrix:
a b[c][d]e f g h
a 1 1 0 0 0 1 0 0
b 1 1 1 0 0 0 0 0
[c] 0 1 1 0 1 1 0 0
[d] 0 0 0 1 1 0 0 0
e 0 0 0 1 1 0 0 0
f 1 0 1 0 0 1 0 0
g 1 0 0 1 0 0 1 0
h 0 0 0 0 0 1 0 1