```c
/* ----------------------------------------------------------- */
/* NAME : Bryan Wandrych                     User ID: bdwandry */
/* DUE DATE : 09/28/2021                                       */
/* PROGRAM ASSIGNMENT 1                                        */
/* FILE NAME : prog1.c                                         */
/* PROGRAM PURPOSE : This programs main purpose is to run      */
/* is to concurrently run multiple different processes at the  */
/* at the same time. These processes are not using shared      */
/* memory and interleaving is expected.                        */
/* ----------------------------------------------------------- */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>

//-----------------------------------------------------------------------------------------------------//
//This section is dedicated towards calculating Fibbonacci sequences.
/* ----------------------------------------------------------- */
/* FUNCTION: Fibbonacci                                        */
/*    To calculate a recurisive number up to N integers        */
/*    Using Fibbonacci sequences.                              */
/* PARAMETER USAGE :                                           */
/*    an integer N for being used in the recursive algorithm   */
/* FUNCTION CALLED :                                           */
/*    StartFibbonacciProcess                                   */
/* ----------------------------------------------------------- */
long Fibbonacci(long n) {
   if(n <= 1){
      return n;
   } else {
      return (Fibbonacci(n - 1) + Fibbonacci(n - 2));
   }
}

/* ----------------------------------------------------------- */
/* FUNCTION: StartFibbonacciProcess                            */
/*    Helper function to be called by a process created        */
/*    This function is also used for printing to stout         */
/* PARAMETER USAGE :                                           */
/*    an integer N for being used in the recursive algorithm   */
/* FUNCTION CALLED :                                           */
/*    main                                                     */
/* ----------------------------------------------------------- */
int StartFibbonacciProcess(long n) {
        char * FibbonacciBuffer = malloc(10000);
        write(1, "   Fibonacci Process Started\n", 29);

        sprintf(FibbonacciBuffer, "   Input Number %ld\n",n);
        write(1, FibbonacciBuffer, strlen(FibbonacciBuffer));
        memset(FibbonacciBuffer, 0, 10000);

        sprintf(FibbonacciBuffer, "   Fibonacci Number f(%d) is %ld\n", n, Fibbonacci(n));
        write(1, FibbonacciBuffer, strlen(FibbonacciBuffer));
        memset(FibbonacciBuffer, 0, 10000);

        write(1, "   Fibonacci Process Exits\n", 27);
}

//-----------------------------------------------------------------------------------------------------//
//This is calculating the Buffon’s Needle Problem
/* ----------------------------------------------------------- */
/* FUNCTION: BuffonsNeedle                                     */
/*    To calculate if a needle will go out of bounds in an     */
/*    in an infinite sqare sequence                            */
/* PARAMETER USAGE :                                           */
/*    an integer r to determine how much a loop runs (iterates)*/
/* FUNCTION CALLED :                                           */
/*    StartBuffonNeedleProcess                                 */
/* ----------------------------------------------------------- */
float BuffonsNeedle (long r) {
        float calculatedTotal = 0.0;
        float L = 1.0;
        float G = 1.0;
        float t = 0.0;

        srand((unsigned) time(NULL));
        for (long i = 0; i < r; i++) {
                float a = (float)rand()/((float)RAND_MAX/(2.0*acos(-1.0)));
                float d = (float)rand()/((float)RAND_MAX/1);
                calculatedTotal = d + L * sin(a);
                if ((calculatedTotal < 0) || (calculatedTotal > G)) {
                        t++;
                }
        }

        return (t/(float)r);
}

/* ----------------------------------------------------------- */
/* FUNCTION: StartBuffonNeedleProcess                          */
/*    Helper function to be called by a process created        */
/*    This function is also used for printing to stout         */
/* PARAMETER USAGE :                                           */
/*    an integer r to determine how much a loop runs (iterates)*/
/* FUNCTION CALLED :                                           */
/*    main                                                     */
/* ----------------------------------------------------------- */
int StartBuffonNeedleProcess(long r) {
        char * BuffonsNeedleBuffer = malloc(10000);
        write(1, "      Buffon’s Needle Process Started\n", 38);

        sprintf(BuffonsNeedleBuffer, "      Input Number %ld\n", r);
        write(1, BuffonsNeedleBuffer, strlen(BuffonsNeedleBuffer));
        memset(BuffonsNeedleBuffer, 0, 10000);

        sprintf(BuffonsNeedleBuffer, "      Estimated Probability is %f\n",BuffonsNeedle(r));
        write(1, BuffonsNeedleBuffer, strlen(BuffonsNeedleBuffer));
        memset(BuffonsNeedleBuffer, 0, 10000);

        write(1, "      Buffon’s Needle Process Exits\n", 36);
}

//-----------------------------------------------------------------------------------------------------//
//This is calculating area of ellipse
/* ----------------------------------------------------------- */
/* FUNCTION: AreaOfEllipse                                     */
/*    To calculate the Area of an Ellipse.                     */
/*    To also watch and see how this calculate, over enough    */
/*    iterations gets closer to Pi*ab                          */
/* PARAMETER USAGE :                                           */
/*    an integer a, to determine a’s value in the equation     */
/*    an integer b, to determine b’s value in the equation     */
/*    an integer s,to determine how much a loop runs (iterates)*/
/* FUNCTION CALLED :                                           */
/*    StartAreaOfEllipseProcess                                */
/* ----------------------------------------------------------- */
float AreaOfEllipse (long a, long b, long s) {
        char * AreaOfEllipseBuffer = malloc(10000);
        float calculation;
        float t = 0.0;
        srand((unsigned) time(NULL));
        for (long i = 0; i < s; i++) {
                float x = (float)rand()/((float)RAND_MAX/(float)a);
                float y = (float)rand()/((float)RAND_MAX/(float)b);
                calculation = (pow(x,2)/pow((float)a,2)) + (pow(y,2)/pow((float)b,2));
                if (calculation <= 1.0) {
                        t++;
                }
        }

        sprintf(AreaOfEllipseBuffer, "         Total Hits %ld\n", (int)t);
        write(1, AreaOfEllipseBuffer, strlen(AreaOfEllipseBuffer));
        memset(AreaOfEllipseBuffer, 0, 10000);
        sprintf(AreaOfEllipseBuffer, "         Estimated Area is %f\n", ((t/s) * (float) a * (float) b)
* 4);
        write(1, AreaOfEllipseBuffer, strlen(AreaOfEllipseBuffer));
        memset(AreaOfEllipseBuffer, 0, 10000);

        sprintf(AreaOfEllipseBuffer, "         Actual Area is %f\n", (acos(-1.0)*a*b));
        write(1, AreaOfEllipseBuffer, strlen(AreaOfEllipseBuffer));
        memset(AreaOfEllipseBuffer, 0, 10000);

        return ((t/s) * (float) a * (float) b) * 4;
}

/* ----------------------------------------------------------- */
/* FUNCTION: StartFibbonacciProcess                              */
/*    Helper function to be called by a process created        */
/*    This function is also used for printing to stout         */
/* PARAMETER USAGE :                                             */
/*    an integer a, to determine a’s value in the equation     */
/*    an integer b, to determine b’s value in the equation     */
/*    an integer s,to determine how much a loop runs (iterates)*/
/* FUNCTION CALLED :                                             */
/*    main                                                       */
/* ----------------------------------------------------------- */
int StartAreaOfEllipseProcess(long a, long b, long s) {
        char * AreaOfEllipseBuffer = malloc(10000);
        write(1, "         Ellipse Area Process Started\n", 38);

        sprintf(AreaOfEllipseBuffer, "         Total random Number Pairs %ld\n", s);
        write(1, AreaOfEllipseBuffer, strlen(AreaOfEllipseBuffer));
        memset(AreaOfEllipseBuffer, 0, 10000);

        sprintf(AreaOfEllipseBuffer, "         Semi-Major Axis Length %ld\n", a);
        write(1, AreaOfEllipseBuffer, strlen(AreaOfEllipseBuffer));
        memset(AreaOfEllipseBuffer, 0, 10000);

        sprintf(AreaOfEllipseBuffer, "         Semi-Minor Axis Length %ld\n", b);
        write(1, AreaOfEllipseBuffer, strlen(AreaOfEllipseBuffer));
        memset(AreaOfEllipseBuffer, 0, 10000);

        AreaOfEllipse(a, b, s);

        write(1, "         Ellipse Area Process Exits\n", 36);
}

//--------------------------------------------------------------------------------------------------//
//This is for the simple pinball game
/* ----------------------------------------------------------- */
/* FUNCTION: PrintArray                                          */
/*    This function is to give a formated printout to Stout    */
/*    Of how many pinballs fell into each "bin"                */
/* PARAMETER USAGE :                                             */
/*    an int [] BinsFilled, this is all balls in bins          */
/*    an integer x, number of bins generated                   */
/*    an integer y,how many balls were dropped into these bings*/
/* FUNCTION CALLED :                                             */
/*    PinballGame                                                */
/* ----------------------------------------------------------- */
 int PrintArray(long BinsFilled [], long x, long y) {
        char * PrintArrayBuffer = malloc(10000);
        float highestPercentage = 0.0;
        long highestPercentageIndex = 0;

        for     (long i = 0; i < x; i++) {
                if ( (((float)BinsFilled[i]/(float)y) * 100) > highestPercentage) {
                        highestPercentage = (((float)BinsFilled[i]/(float)y) * 100);
                        highestPercentageIndex = i;
                }
        }

        for (long i = 0; i < x; i++) {
                sprintf(PrintArrayBuffer, "%3d-(%7ld)-(%5.2f%)│", (i + 1), BinsFilled[i],
((float)BinsFilled[i]/(float)y) * 100);
                int NumOfAsterix = round(((((float)BinsFilled[i]/(float)y) * 100)/highestPercentage) *
50);
```

```
                for (int j = 0; j < NumOfAsterix; j++) {
                        sprintf(PrintArrayBuffer + strlen(PrintArrayBuffer), "*");
                }
                sprintf(PrintArrayBuffer + strlen(PrintArrayBuffer), "\n");
                write(1, PrintArrayBuffer, strlen(PrintArrayBuffer));
                memset(PrintArrayBuffer, 0, 10000);
        }
}

/* ----------------------------------------------------------- */
/* FUNCTION: PinballGame                                         */
/*    This is the function that will randomly sort the balls   */
/*    into n number of generated bins                          */
/* PARAMETER USAGE :                                             */
/*    an int [] BinsFilled, this is all balls in bins          */
/*    an integer x, number of bins generated                   */
/*    an integer y,how many balls were dropped into these bings*/
/* FUNCTION CALLED :                                             */
/*    StartPinballGameProcess                                    */
/* ----------------------------------------------------------- */
int PinballGame (long x, long y) {
        long BinsFilled [x];
        for (int i = 0; i < x; i++) {
                BinsFilled[i] = 0;
        }

        srand((unsigned) time(NULL));
        for (long i = 0; i < y; i++) {
                long ballDirection = 0;

                for (long j = 0; j < x-1; j++) {
                        float random = (float)rand()/((float)RAND_MAX/1);
                        if (random >= .5) {
                                ballDirection++;
                        }
                }
                BinsFilled[ballDirection]++;
        }
        PrintArray(BinsFilled, x, y);
        return 1;
}

/* ----------------------------------------------------------- */
/* FUNCTION: StartPinballGameProcess                             */
/*    Helper function to be called by a process created        */
/*    This function is also used for printing to stout         */
/* PARAMETER USAGE :                                             */
/*    an int [] BinsFilled, this is all balls in bins          */
/*    an integer x, number of bins generated                   */
/*    an integer y,how many balls were dropped into these bings*/
/* FUNCTION CALLED :                                             */
/*    main                                                       */
/* ----------------------------------------------------------- */
int StartPinballGameProcess(long x, long y) {
        char * PinballGameBuffer = malloc(10000);
        write(1, "Simple Pinball Process Started\n", 31);

        sprintf(PinballGameBuffer, "Number of Bins %ld\n", x);
        write(1, PinballGameBuffer, strlen(PinballGameBuffer));
        memset(PinballGameBuffer, 0, 10000);

        sprintf(PinballGameBuffer, "Number of Ball Droppings %ld\n", y);
        write(1, PinballGameBuffer, strlen(PinballGameBuffer));
        memset(PinballGameBuffer, 0, 10000);

        PinballGame(x,y);

        write(1, "Simple Pinball Process Exits\n", 29);
}


//--------------------------------------------------------------------------------------------------//
//This is the main function
/* ----------------------------------------------------------- */
/* FUNCTION: main                                               */
/*    This function is the main function to this program       */
/*    It will be the starting point for the program as well as */
/*    Spawn all of the extra processes                          */
/* PARAMETER USAGE :                                            */
/*    argc and argv to pass various arguments through the      */
/*    command line once the program is compiled                 */
/* FUNCTION CALLED :                                            */
/*    N/A                                                       */
/* ------------------------------------------------------------ */
int main(int argc, char *argv[]) {
        if (argc != 8) {
                printf("./prog1 n r a b s x y\n");
        } else {
                //Main Process Starting
                char * MainProcessBuffer = malloc(10000);
                write(1, "Main Process Started\n", 21);

                sprintf(MainProcessBuffer, "Fibonacci Input            = %ld\n", atol(argv[1]));
                write(1, MainProcessBuffer, strlen(MainProcessBuffer));
                memset(MainProcessBuffer, 0, 10000);

                sprintf(MainProcessBuffer, "Buffon’s Needle Iterations = %ld\n", atol(argv[2]));
                write(1, MainProcessBuffer, strlen(MainProcessBuffer));
                memset(MainProcessBuffer, 0, 10000);

                sprintf(MainProcessBuffer, "Total random Number Pairs  = %ld\n", atol(argv[3]));
                write(1, MainProcessBuffer, strlen(MainProcessBuffer));
                memset(MainProcessBuffer, 0, 10000);

                sprintf(MainProcessBuffer, "Semi-Major Axis Length     = %ld\n", atol(argv[4]));
                write(1, MainProcessBuffer, strlen(MainProcessBuffer));
                memset(MainProcessBuffer, 0, 10000);

                sprintf(MainProcessBuffer, "Semi-Minor Axis Length     = %ld\n", atol(argv[5]));
                write(1, MainProcessBuffer, strlen(MainProcessBuffer));
                memset(MainProcessBuffer, 0, 10000);

                sprintf(MainProcessBuffer, "Number of Bins             = %ld\n", atol(argv[6]));
                write(1, MainProcessBuffer, strlen(MainProcessBuffer));
                memset(MainProcessBuffer, 0, 10000);

                sprintf(MainProcessBuffer, "Number of Ball Droppings   = %ld\n", atol(argv[7]));
                write(1, MainProcessBuffer, strlen(MainProcessBuffer));
                memset(MainProcessBuffer, 0, 10000);

                pid_t pid[4];
                //Fibbonacci
                if ((pid[0] = fork()) == 0) {
                        write(1, "Fibonacci Process Created\n", 26);
                        StartFibbonacciProcess(atol(argv[1]));

                        exit(0);
                }

                //BuffonsNeedle
                if ((pid[1] = fork()) == 0) {
                        write(1, "Buffon’s Needle Process Created\n", 32);
                        StartBuffonNeedleProcess(atol(argv[2]));

                        exit(0);
                }

                //AreaOfEllipse
                if ((pid[2] = fork()) == 0) {
                        write(1, "Ellipse Area Process Created\n", 29);
                        StartAreaOfEllipseProcess(atol(argv[3]), atol(argv[4]), atol(argv[5]));

                        exit(0);
                }

                //PinbalGame
                if ((pid[3] = fork()) == 0) {
                        write(1, "Pinball Process Created\n", 24);
                        StartPinballGameProcess(atol(argv[6]), atol(argv[7]));

                        exit(0);
                }

                //Main is now Waiting
                int status;
                write(1, "Main Process Waits\n", 19);

                for (int i = 0; i < 4; i++) {
                        wait(&status);
                }

                //Main is now Exiting
                write(1, "Main Process Exits\n", 19);
        }

        return 1;

}
```

```
============================COMPILATION================================
Compilation done.
============================TEST 1=====================================
Main Process Started
Fibonacci Input            = 10
Buffon’s Needle Iterations = 100000
Total random Number Pairs  = 6
Semi-Major Axis Length     = 2
Semi-Minor Axis Length     = 200000
Number of Bins             = 6
Number of Ball Droppings   = 3000000
Fibonacci Process Created
   Fibonacci Process Started
Buffon’s Needle Process Created
   Input Number 10
   Fibonacci Number f(10) is 55
   Fibonacci Process Exits
      Buffon’s Needle Process Started
Main Process Waits
      Input Number 100000
Ellipse Area Process Created
         Ellipse Area Process Started
         Total random Number Pairs 200000
Pinball Process Created
         Semi-Major Axis Length 6
         Semi-Minor Axis Length 2
Simple Pinball Process Started
Number of Bins 6
Number of Ball Droppings 3000000
      Estimated Probability is 0.634180
      Buffon’s Needle Process Exits
         Total Hits 156960
         Estimated Area is 37.670399
         Actual Area is 37.699112
         Ellipse Area Process Exits
  1-(  94015)-( 3.13%)|*****
  2-( 468271)-(15.61%)|*************************
  3-( 938239)-(31.27%)|**************************************************
  4-( 937354)-(31.25%)|**************************************************
  5-( 468294)-(15.61%)|*************************
  6-(  93827)-( 3.13%)|*****
Simple Pinball Process Exits
Main Process Exits
============================TEST 2=====================================

Main Process Started
Fibonacci Input            = 11
Buffon’s Needle Iterations = 200000
Total random Number Pairs  = 7
Semi-Major Axis Length     = 3
Semi-Minor Axis Length     = 300000
Number of Bins             = 8
Number of Ball Droppings   = 4000000
Fibonacci Process Created
   Fibonacci Process Started
Buffon’s Needle Process Created
   Input Number 11
   Fibonacci Number f(11) is 89
   Fibonacci Process Exits
      Buffon’s Needle Process Started
Main Process Waits
Ellipse Area Process Created
      Input Number 200000
         Ellipse Area Process Started
Pinball Process Created
         Total random Number Pairs 300000
Simple Pinball Process Started
         Semi-Major Axis Length 7
         Semi-Minor Axis Length 3
Number of Bins 8
Number of Ball Droppings 4000000
      Estimated Probability is 0.636400
      Buffon’s Needle Process Exits
         Total Hits 235573
         Estimated Area is 65.960434
         Actual Area is 65.973446
         Ellipse Area Process Exits
  1-(  31382)-( 0.78%)|*
  2-( 218416)-( 5.46%)|**********
  3-( 656339)-(16.41%)|******************************
  4-(1094682)-(27.37%)|**************************************************
  5-(1093211)-(27.33%)|**************************************************
  6-( 655775)-(16.39%)|******************************
  7-( 219081)-( 5.48%)|**********
  8-(  31114)-( 0.78%)|*
Simple Pinball Process Exits
Main Process Exits
```

Bryan Wandrych
bdwandry
bdwandry@mtu.edu
M17571110
9/29/21

Note Please view in a text editor that doesn’t have text wrapping enabled by default. Or else the diagrams are gonna look messed up.
Notepad++ is a good example of a text editor to view this page.

1. Question: Draw a diagram showing the parent-child relationship if the following program is run with command line argument 4.
How many processes are created? Explain step-by-step how these processes are created, especially who is created by whom.

```
void main(int argc, char **argv) {
        int i, n = atoi(argv[1]);
        for (i = 1; i < n; i++)
                if (fork())
                        break;
        printf("Process %ld with parent %ld\n", getpid(), getppid());
        sleep(1);
}
```

✓

Answer:

P—C—C—C

```
A=Parent Process
B=Child Process

 â\224\214â\224\200â\224\200â\224\200â\224\220
 â\224¤ A â\224\234
 â\224\234â\224\200â\224\200â\224\200â\224¤
   â\224\202
   â\224\202
   â\224\202
 â\224\214â\224\200â\226¼â\224\200â\224\220
 â\224\202B_1â\224\202
 â\224\224â\224\200â\224\200â\224\200â\224\230
   â\224\202
   â\224\202
   â\224\202
 â\224\214â\224\200â\226¼â\224\200â\224\220
 â\224\202c_1â\224\202
 â\224\224â\224\200â\224\200â\224\200â\224\230
   â\224\202
  ...
   â\224\202
 â\224\214â\224\200â\226¼â\224\200â\224\200â\224\200â\224\220
 â\224\202n_n-1â\224\202
 â\224\224â\224\200â\224\200â\224\200â\224\200â\224\200â\224\230
```

Demonstrated above in the diagram. This forking tree will always become a straight line down starting at the parent process, this is because the parent process
will fork to the child, and then the program breaks out of the loop, then the child process becomes a parent to another child process. This again will happen
N number of times.

2. Draw a diagram showing the parent-child relationship if the following program is run with command line argument 4.
How many processes are created? Explain step-by-step how these processes are created, especially who is created by whom.

```
void main(int argc, char **argv) {
        int i, n = atoi(argv[1]);
        for (i = 0; i < n; i++)
                if (fork() <= 0)
                        break;
        printf("Process %ld with parent %ld\n", getpid(), getppid());
        sleep(1);
{
```

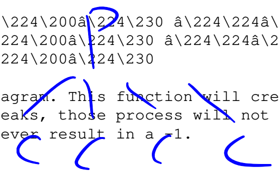
Answer:

```
           A=Parent Process
           B=Child Process

               â\224\214â\224\200â\224\200â\224\200â\224\220

â\224\214â\224\200â\224\200â\224\200â\224\200â\224\200â\224¬â\224\200â\224\200â\224\200â\224\200â\224
\200â\224¤ A
â\224\234â\224\200â\224¬â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224
\200â\224\220
   â\224\202     â\224\202     â\224\234â\224\200â\224\200â\224\200â\224¤ â\224\202         â\224\202
   â\224\202     â\224\202     â\224\202     â\224\202          â\224\202
   â\224\202     â\224\202     â\224\202     â\224\202          â\224\202
   â\224\202     â\224\202     â\224\202     â\224\202          â\224\202
 â\224\214â\224\200â\226¼â\224\200â\224\220 â\224\214â\224\200â\226¼â\224\200â\224\220
â\224\214â\224\200â\226¼â\224\200â\224\220 â\224\214â\224\200â\226¼â\224\200â\224\220
â\224\214â\224\200â\226¼â\224\200â\224\220
 â\224\202B_1â\224\202 â\224\202B_2â\224\202 â\224\202B_3â\224\202 â\224\202B_4â\224\202....
â\224\202B_nâ\224\202
 â\224\224â\224\200â\224\200â\224\200â\224\230 â\224\224â\224\200â\224\200â\224\200â\224\230
â\224\224â\224\200â\224\200â\224\200â\224\230 â\224\224â\224\200â\224\200â\224\200â\224\230
â\224\224â\224\200â\224\200â\224\200â\224\230
```

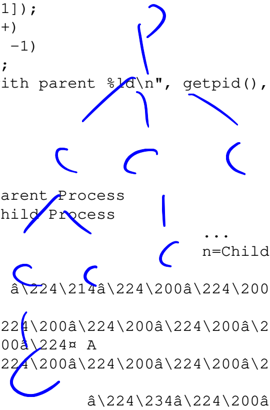
✓



Demonstrated in the above diagram. This function will create N number of children for the original parent process. Because it breaks, those process will not go any further and it
is very rare for forking to ever result in a -1.

3. Draw a diagram showing the parent-child relationship if the following program is run with command line argument 3.
How many processes are created? Explain step-by-step how these processes are created, especially who is created by whom.

```
void main(int argc, char **argv) {
        int i, n = atoi(argv[1]);
        for (i = 0; i < n; i++)
                if (fork() == -1)
                        break;
        printf("Process %ld with parent %ld\n", getpid(), getppid());
        sleep(1);
}
```



✓

Answer:

```
                         A=Parent Process
                         B=Child Process
                                                 ...
                                                 n=Child Process

                              â\224\214â\224\200â\224\200â\224\200â\224\220

â\224\214â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224
\200â\224\200â\224\200â\224\200â\224¤ A
â\224\234â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224\200â\224
\200â\224\220
               â\224\202           â\224\234â\224\200â\224\200â\224\200â\224\230           â\224\202
               â\224\202           â\224\202            â\224\202
               â\224\202           â\224\202            â\224\202
               â\226¼         â\224\202            â\224\202
             â\224\214â\224\200â\226¼â\224\200â\224\220
â\224\214â\224\200â\226¼â\224\200â\224\220            â\224\214â\224\200â\226¼â\224\200â\224\220
             â\224\202B_1â\224\202           â\224\202B_2â\224\202    ....   â\224\202B_nâ\224\202
            â\224\200â\224´â\224\200â\224\200â\224\200â\224´â\224\220
â\224\200â\224´â\224\200â\224\200â\224\200â\224´           â\224\224â\224\200â\224\200â\224\200â\224\230
           â\224\202     â\224\202     â\224\202
           â\226¼     â\226¼     â\226¼
         â\224\214â\224\200â\226¼â\224\200â\224\220   â\224\214â\224\200â\227\204â\224\200â\224\220
â\224\214â\224\200â\226¼â\224\200â\224\220
         â\224\202c_1â\224\202   â\224\202c_2â\224\202  â\224\202c_3â\224\202
     â\224\214â\224\200â\224\200â\224\200â\224´â\224\200â\224\200â\224\200â\224\230
```

```
â\224\224â\224\200â\224\200â\224\200â\224\230  â\224\224â\224\200â\224\200â\224\200â\224\230
     â\224\202            ...    ...
     â\224\202
   â\224\214â\224\200â\226¼â\224\200â\224\220
   â\224\202d_1â\224\202
   â\224\224â\224\200â\224\200â\224\200â\224\230
    ...
```

This process tree will create a fractal pattern in which all left process are guarenteed to have children. But as you start moving left to right in the forking tree.
The process to the right of the left process will have n-1 children. Thus the last process being created will have 0 children.

4.The histogram you obtained from the simple pinball game is always symmetric, even though the number of balls in each bin may be slightly different.
However, if the histogram is significantly not symmetric, your program is definitely incorrect.
Actually, this is a distribution you may have learned in your statistics and probability course. What is this distribution called?
What is the reason you believe the histogram is the named distribution by you? Answer this question with a good logic reasoning.
Without doing so (e.g., only writing done the answer with a vague reason), you will lose point for this portion.

Example from my program:

```
   Bins: 6
   Balls: 30000000

  1-( 936602)-( 3.12%)|*****
  2-(4683225)-(15.61%)|*************************
  3-(9377423)-(31.26%)|***************************************************
  4-(9373490)-(31.24%)|***************************************************
  5-(4689902)-(15.63%)|*************************
  6-( 939358)-( 3.13%)|*****
```

✓

Answer:
What is this distribution called?
        This would be known as a binomial distrubution.

What is the reason you believe the histogram is the named distribution by you?
        This is a binomial distribution due to the symmetry of the data being recorded and the lack of continuous datapoints (meaning you can’t land in between two bins (a ball can’t land in 3.5 bins its only 3 or 4 for bin number)).

CS3331 Program I Grade Report

You receive 0 point if any one of the following occurs

No further grading will be done

Problem	*Check All Apply*	*You Receive*
Not-compile		0
Compile-but-not-run		0
Meaningless and/or vague Program		0
Did not implement the indicated methods		0
Did not follow the required program structure		0
Other significant deviation from specification		0
Totally wrong and unacceptable output		0

This part applies only if you have a working program

Item		*Max Possible*	*You Receuve*
Style & Doc.	Header in each file	1	1
	Good indentation	1	1
	Good comments	1	1
	Good use of function, variable names, etc & no `GOTO`	1	1
Spec	Handles command line input properly	2	2
	Correct output format	2	2
Correctness	Work on sample data	17	17
	Work on our data	17	17
`README`	Missing `README` – next two items receive 0	0	0
	Well-written `README`	3	3
	Answer questions properly	5	5
Total		50	50

Your Random Number:________ **Your Score:** 50