NAME : RAHUL THAPAR
ID : 1410110321

Submit the following entries in a word file:

Problem Statement:
Problem Statement: How to simulate an n-sided coin using a 2 sided coin. (Solve for n=6).

Algorithm:
Solution: You can simulate an n-sided coin using a two sided coin as follows:

Let m = $\lceil \log n \rceil$. The base is always 2. (Example, for n = 6, m = 3)

Flip a 2-sided coin m times and record the result of every flip. (HHT may be represented as 110)

Convert the binary number generated to a decimal number. (Example: $(110)_2 = (6)_{10}$)

Repeat for the number of sample points required.

Challenge:

If m =3, the numbers generated will be in the range 0 to 7, whereas we need the numbers in the range (1, 6).

A possible Solution-

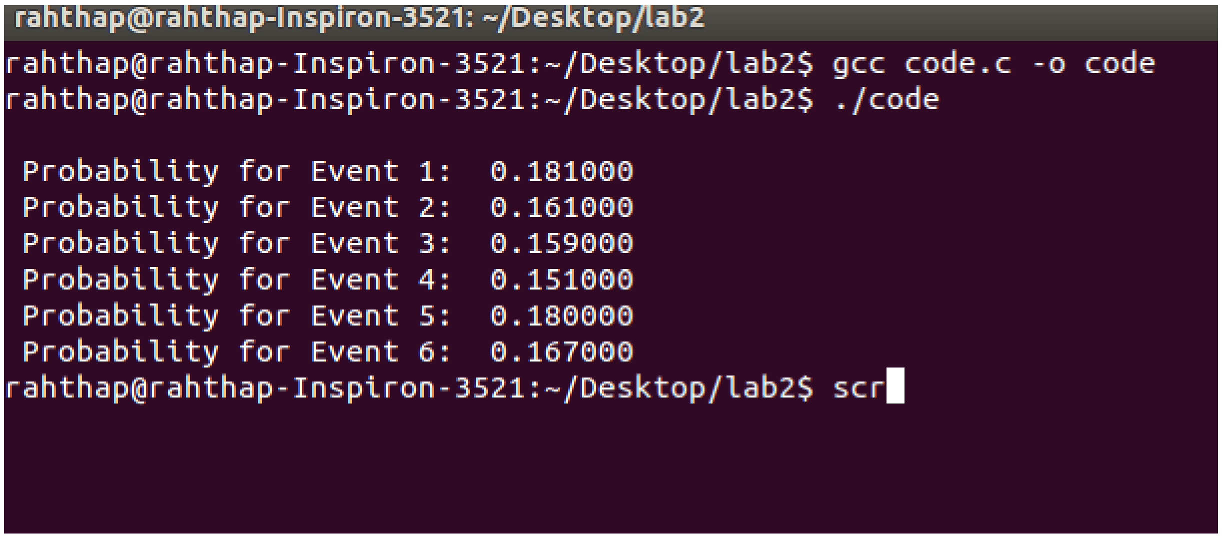
- When you get a number not in range, ignore it and regenerate another number in range.
  In this example – When you generate a 0 or a 7, ignore it and generate another number till you get a number in the range and record that.

Note: When n = 6, we can simulate a dice using a 2-sided coin.

C Code

```
/***
_
  NAME : Rahul Thapar
  ID : 1410110321
  DATE : 18-01-2017

***/


#include<stdio.h>
#include<stdlib.h>
#define MAX 1000

void main(){
        srand(time(NULL));
        int i;
        int events[MAX];
        int outcome_1, outcome_2, outcome_3;
        int number =0;
        int f1=0,f2=0,f3=0,f4=0,f5=0,f6=0;
        float p1, p2, p3, p4, p5, p6;

        for(i=0;i<MAX;i++){
                outcome_1 = rand()%2;
        outcome_2 = rand()%2;
                outcome_3 = rand()%2;

                number = (outcome_1*4)+(outcome_2*2)+outcome_3;


        while(number==0||number==7)
    {
       outcome_1 = rand()%2;
       outcome_2 = rand()%2;
       outcome_3 = rand()%2;
       number = (outcome_1*4)+(outcome_2*2)+outcome_3;

    }
   events[i]=number;
}

        for(i=1;i<=MAX;i++){
           if(events[i]==1)
         f1++;
        if(events[i]==2)
                f2++;
        if(events[i]==3)
                f3++;
                 if(events[i]==4)
                f4++;
                 if(events[i]==5)
                f5++;
        if(events[i]==6)
                f6++;
 }
 printf("\n Probability for Event 1:  %f ",(float)f1/1000);
 printf("\n Probability for Event 2:  %f ",(float)f2/1000);
 printf("\n Probability for Event 3:  %f ",(float)f3/1000);
 printf("\n Probability for Event 4:  %f ",(float)f4/1000);
 printf("\n Probability for Event 5:  %f ",(float)f5/1000);
 printf("\n Probability for Event 6:  %f ",(float)f6/1000);
 printf("\n");
}
```

Result Table
Sample space is the following: {1, 2, 3, 4, 5, 6}
Find the probability of each event, while generating 1000 samples points.



```
rahthap@rahthap-Inspiron-3521: ~/Desktop/lab2
rahthap@rahthap-Inspiron-3521:~/Desktop/lab2$ gcc code.c -o code
rahthap@rahthap-Inspiron-3521:~/Desktop/lab2$ ./code

 Probability for Event 1:  0.181000
 Probability for Event 2:  0.161000
 Probability for Event 3:  0.159000
 Probability for Event 4:  0.151000
 Probability for Event 5:  0.180000
 Probability for Event 6:  0.167000
rahthap@rahthap-Inspiron-3521:~/Desktop/lab2$ scr
```

Event	Probability of event
1	0.181000
2	0.161000
3	0.159000
4	0.151000
5	0.180000
6	0.167000

Analysis
Did the result meet the expectation?

No, the results obtained did not meet the expectations. The probability for each event should have been same but it is not the case.

If no, can you think of an improvement?

The algorithm can be further optimized.