```cpp
#include <fstream>
#include <string>
#include <iostream>
#include <random>
#include "engine.h"
#include "constants.h"

using namespace std;
static random_device rdev;
static default_random_engine engine (rdev());


void loadBlockData(string gameFile, int & numBlocks, int blockPosX[], int blockPosY[],
   bool blockTop[])
{

    //Declare and open filestream
    fstream fin;
    fin.open(gameFile.c_str(),ios::in);

    //Read in Number of blocks

    int totalColumnCount = 0;
    int colCount = 0;
    int rowCount = 0;

    while(totalColumnCount<SCREEN_WIDTH/BLOCK_SPRITE_WIDTH)
    {

        fin >> colCount >> rowCount;

        for(int i=0;i<colCount;i++)
        {

            for(int j=0;j<rowCount;j++)
            {
                blockPosX[numBlocks] = totalColumnCount*BLOCK_SPRITE_WIDTH;
                blockPosY[numBlocks] = SCREEN_HEIGHT-(j+1)*BLOCK_SPRITE_HEIGHT;
                if(j==(rowCount-1))
                {
                    blockTop[numBlocks] = true;
                }
                else
                {
                    blockTop[numBlocks] = false;
                }
                numBlocks++;
            }
            totalColumnCount++;
        }

    }
```

```
    //Close the stream
    fin.close();
}

void randomZelda2Data(int & linkPosX, int & linkPosY, int & linkSpriteID, int
  numBlocks, int blockPosX[], int blockPosY[], bool blockTop[])
{
    static uniform_int_distribution<int> rLinkX(0, SCREEN_WIDTH - Z2_SPRITE_WIDTH);
    static uniform_int_distribution<int> rLinkSprite(0, Z2_NUM_SPRITES-1);

    //Pick a random center x location for the sprite.
    linkPosX = rLinkX(engine);
    //Figure out what the highest height below the sprite would be.
    linkPosY = getMaxYOfBlock(linkPosX, Z2_SPRITE_WIDTH, numBlocks,blockPosX,
      blockPosY, blockTop);
    //Pick a random sprite from the sprite sheet
    linkSpriteID = rLinkSprite(engine);


}


int getMaxYOfBlock(int x, int width, int numBlocks, int blockPosX[], int blockPosY[],
  bool blockTop[])
{

    int xStart = x;
    int xEnd = xStart+width;

    //Set Lowest Position Possible
    int maxY = SCREEN_HEIGHT;

    //Refine miny based on overlap with blocks
    for(int i=0;i<numBlocks;i++)
    {

        //Only check the top block's positions
        if(blockTop[i])
        {

            //Calculate 'this' block's start/end
            int bStart = blockPosX[i];
            int bEnd = blockPosX[i]+BLOCK_SPRITE_WIDTH;

            //Occlude pattern 1
            if(bEnd>= xStart && bStart<xStart)
            {
                if(blockPosY[i]<maxY)
                {
                    maxY = blockPosY[i];
                }
```

```
            }

            //Occlude pattern 2
            if(bStart>=xStart && bEnd<= xEnd)
            {
                if(blockPosY[i]<maxY)
                {
                    maxY = blockPosY[i];
                }
            }

            //Occlude pattern 3
            if(bEnd>=xEnd && bStart<xEnd)
            {
                if(blockPosY[i]<maxY)
                {
                    maxY = blockPosY[i];
                }
            }
        }
    }

    return(maxY-Z2_SPRITE_HEIGHT);

}
```