前言

~~显然这是量子计算的模型。（当然有可能有我理解不正确的地方。）显然题目中没有出现相关算法，只是量子计算入门。只要熟悉了就能很快写完。除了阅读理解外是一个签到题。~~

~~命题想法当然是想到了题目末尾提到的东西，应该可以作为一个量子计算的小科普，真正想学算法可以自行检索。~~

以下“观测”/“测量”均表示题目中的“读取”。

任务1

我们想要有$p = 14343375/16777216$的概率得到 1，其余得到 0，就可以用`ReturnFull`来得到想要的生成器。而可以生成 $\sqrt{1-p}|0\rangle + \sqrt{p}|0\rangle$ 并测量它可以做到这一点。而 `Ry` 可以看作对 (x, y) 逆时针转 $\theta/2$，其中 x, y 指示了状态 $x|0\rangle + y|0\rangle$，故可以通过旋转 $\arcsin(\sqrt{p})$ 来做到这一点。

```
void solve01(const quantum::Printer &ot)
{
    ot.Begin();
    ot.Ry(0, asinl(sqrtl(0xdadccf / 16777216.)) * 2);
    ot.M(0);
    ot.ReturnFull({0, 0xdadccf});
    ot.End();
}
```

任务2

由于无论是 $|0\rangle$ 还是 $|1\rangle$，经过 `H` 后为 $(|0\rangle \pm |1\rangle)/\sqrt{2}$，而它被观测后有 $1/2$ 的概率返回 0，$1/2$ 的概率返回 1，然后变为 $|0\rangle$ 或 $|1\rangle$。故可连续 `H` 后测量两次得到结果。

```
void solve02(const quantum::Printer &ot)
{
    ot.Begin();
    ot.H(0);
    ot.M(0);
    ot.H(0);
    ot.M(0);
    ot.ReturnFull({1, 2, 3, 4});
    ot.End();
}
```

任务3

原理同任务1，然后用分支套个娃即可。

```
void solve03(const quantum::Printer &ot)
{
    // ot.Begin();
    // ot.Ry(0, asinl(sqrtl(0.3+0.4)) * 2);
    // ot.IfM(0);
    // {
    //     ot.Ry(0, -quantum::pi+asinl(sqrtl(0.3 / (0.3 + 0.4))) * 2);
    //     ot.IfM(0), ot.Return(3), ot.Else(), ot.Return(4), ot.Endif();
    // }
    // ot.Else();
    // {
    //     ot.Ry(0, asinl(sqrtl(0.1 / (0.1 + 0.2))) * 2);
    //     ot.IfM(0), ot.Return(1), ot.Else(), ot.Return(2), ot.Endif();
    // }
    // ot.Endif();
```

```
    // ot.End();

    ot.Begin();
    ot.Ry(0, asinl(sqrtl(0.1)) * 2);
    ot.IfM(0), ot.Return(1), ot.Else();
    ot.Ry(0, asinl(sqrtl(0.2 / 0.9)) * 2);
    ot.IfM(0), ot.Return(2), ot.Else();
    ot.Ry(0, asinl(sqrtl(0.3 / 0.7)) * 2);
    ot.IfM(0), ot.Return(3), ot.Else();
    ot.Return(4);
    ot.Endif();
    ot.Endif();
    ot.Endif();
    ot.End();
}
```

任务4

同任务1，可以先生成一个 $1/5$ 概率返回1的比特，然后测量它，再用任务2的方法连续测两次，就可以得到 $1/20, 1/20, 1/20, 1/20, 1/5, 1/5, 1/5, 1/5$的8种情况的概率分布，然后分配一下即可。

```
void solve04(const quantum::Printer &ot)
{
    ot.Begin();
    ot.Ry(0, asinl(sqrtl(0.2)) * 2);
    ot.M(0), ot.H(0), ot.M(0), ot.H(0), ot.M(0);
    ot.ReturnFull({2, 1, 3, 1, 4, 3, 4, 3});
    ot.End();
}
```

任务5

可以先用 `Ry` 调对两个系数的模长，再用 `Rz` 调对两个系数在复平面之间的夹角，注意只用调这两个，因为可以任意乘 $e^{i\theta}$ 把他们转到正确的位置。

```
void solve05(const quantum::Printer &ot)
{
    ot.Begin();
    ot.Ry(0, asinl(sqrtl(0.7)) * 2), ot.Rz(0, -atanl(sqrtl(2)) + atanl(sqrtl(4. / 3)));
    ot.End();
}
```

任务6

可以通过连续两次 `T` 将 $|1\rangle$ 虚数上的系数调到实数上，变成 $(|0\rangle \pm |1\rangle)/\sqrt{2}$，然后再用下面的例子中的提示，通过一个 `H` 即可调成 $|0\rangle$ 或 $|1\rangle$，再测量即可。

```
void solve06(const quantum::Printer &ot)
{
    ot.Begin();
    ot.T(0), ot.T(0), ot.H(0);
    ot.M(0);
    ot.ReturnFull({1, 0});
    ot.End();
}
```

任务7

实际上如果只有 $|0\rangle$ 或 $|1\rangle$ 之一有系数，显然是做不到这一点的，因此我们可以先作用一个 `H` 变为叠加态，然后进行测试。如果没有操作，将会得到 $(|0\rangle+|1\rangle)/\sqrt{2}$；否则，有 $1/2$ 的概率变成 1，然后炸了，否则一定为 $|0\rangle$，这时候如果我们再次作用 `H`，那么当没有炸弹时，会变成 $|0\rangle$，有炸弹且没炸时会变成 $(|0\rangle+|1\rangle)/\sqrt{2}$，这时候再测量，当没有炸弹的时候，测量结果一定是0；有炸弹的时候测量结果有 $1/2$ 的概率是1。故如果是炸弹，则有 $1/2$ 的概率爆炸，$1/4$ 的概率测量结果是 1；没有炸弹时测量结果一定是 0，这样就满足题目要求了。

```
void solve07(const quantum::Printer &ot)
{
    ot.Begin();
    ot.H(0);
    ot.End();

    ot.Begin();
    ot.H(0);
    ot.M(0);
    ot.ReturnFull({-1, 1});
    ot.End();
}
```

任务8

$(|00\rangle+i|10\rangle+i|01\rangle-|11\rangle)/2=(|0\rangle+i|1\rangle)/\sqrt{2}\otimes(|0\rangle+i|1\rangle)/\sqrt{2}$，只要对两个比特分别生成 $(|0\rangle+i|1\rangle)/\sqrt{2}$ 即可，这可以通过 `Rx` 做到。

```
void solve08(const quantum::Printer &ot)
{
    ot.Begin();
    ot.Rx(0, -quantum::pi / 2), ot.Rx(1, -quantum::pi / 2);
    ot.End();
}
```

任务9

熟知 `x^=y^=x^=y` 可以对两个数进行交换，而当 `x[0]` 为控制比特，用 `X` 作用 `x[1]` 时，相当于 `x[1]^=x[0]`。

```
void solve09(const quantum::Printer &ot)
{
    ot.Begin();
    ot.X(1, 1), ot.X(0, 2), ot.X(1, 1);
    ot.End();
}
```

任务10

可以有很多做法，比如先对比特0用 `X` 生成 $|10\rangle$ 再用 `H` 生成 $(|00\rangle-|10\rangle)/\sqrt{2}$，再以比特0为控制比特，对比特1进行 `X`，可以得到 $(|00\rangle-|11\rangle)/\sqrt{2}$，再对比特0用 `X` 即得 $(|10\rangle-|01\rangle)/\sqrt{2}$

```
void solve10(const quantum::Printer &ot)
{
    ot.Begin();
    ot.X(0), ot.H(0), ot.X(1, 1), ot.X(0);
    ot.End();
}
```

任务11

可以先用 `Ry` 调比特0的0和1的模长（$|x0\rangle$，$|x1\rangle$ 系数模长之和），然后以比特0位控制比特，旋转比特1，得到当比特0为1时的系数；把比特0用 `X` 取反后再同样操作，得到当比特0为0时的系数，最后再将比特0取反。可以通过调整使得最后对比特0进行 `X` 的操作可以不做。（就是先把最终系数中比特0取反。）

```
void solve11(const quantum::Printer &ot)
{
    ot.Begin();
    ot.Ry(1, asinl(sqrtl(.3)) * 2), ot.Ry(0, asinl(sqrtl(.2 / .3)) * 2, 2),
        ot.X(1), ot.Ry(0, asinl(sqrtl(.4 / .7)) * 2, 2);
    ot.End();
}
```

任务12

可以通过前面控制后面的旋转，使得初始为 10 时相邻的得到 10 和 11 按一定比例，且初始为 00 时不影响。再后面控制前面取反可以得到一定比例的 10 和 01，然后递推地做即可。

```
void solve12(const quantum::Printer &ot)
{
    int32_t i;
    ot.Begin();
    ot.X(0);
    for (i = 0; i < 6; i++)
        ot.Ry(i + 1, asinl(sqrtl((6. - i) / (7. - i))) * 2, 1 << i), ot.X(i, 1 << (i + 1));
    ot.End();
}
```

任务13

由于不允许旋转，可以先生成 $|10000000\rangle$，然后用对比特1的 `H` 生成 $(|10000000\rangle+|11000000\rangle)/\sqrt{2}$，再用比特1控制比特0的 `X` 可以得到 $(|10000000\rangle+|01000000\rangle)/\sqrt{2}$，然后对比特4 `H`，用比特4控制比特0、2，比特1、3交换，得到 $(|10000000\rangle+|01000000\rangle+|00101000\rangle+|00011000\rangle)/\sqrt{4}$，再分别以比特2、3控制比特 4 的 `X` 消除比特4的影响，得到 $(|10000000\rangle+|01000000\rangle+|00100000\rangle+|00010000\rangle)/\sqrt{4}$。再对比特 4 用 `H` 后控制比特1、5，比特2、6，比特3、7交换，得到 $(|10000000\rangle+|01000000\rangle+|00100000\rangle+|00010000\rangle+|10001000\rangle+|00001100\rangle+|00001010\rangle+|00001001\rangle)/\sqrt{8}$，再分别以比特5、6、7控制比特 4 的 `X` 消除比特4的影响，得到 $(|10000000\rangle+|01000000\rangle+|00100000\rangle+|00010000\rangle+|10001000\rangle+|00000100\rangle+|00000010\rangle+|00000001\rangle)/\sqrt{8}$，最后用比特4控制比特0的 `X` 即可。

```
void solve13(const quantum::Printer &ot)
{
    ot.Begin();
    ot.X(0), ot.H(1), ot.X(0, 2);
    ot.H(4), ot.SWAP(0, 2, 1 << 4), ot.SWAP(1, 3, 1 << 4), ot.X(4, 1 << 2), ot.X(4, 1 << 3);
    ot.H(4), ot.SWAP(1, 5, 1 << 4), ot.SWAP(2, 6, 1 << 4), ot.SWAP(3, 7, 1 << 4);
    ot.X(4, 1 << 5), ot.X(4, 1 << 6), ot.X(4, 1 << 7), ot.X(0, 1 << 4);
    ot.End();
}
```

任务14

同任务11，一步步做即可。

```
int C[6][6];
void dfs(const quantum::Printer &ot, int cur, int cnt)
{
    if (cnt == 2)
```

```
        return;
    assert(cur<5);
    if (5 - cur + cnt == 2)
    {
        ot.X(cur, (1 << cur) - 1);
        dfs(ot, cur + 1, cnt + 1);
        return;
    }
    ot.Ry(cur, atan2l(sqrtl(C[4 - cur][2 - cnt]), sqrtl(C[4 - cur][1 - cnt])) * 2, (1 << cur) -
1);
    // 0
    dfs(ot,cur+1,cnt);
    // 1
    ot.X(cur,(1<<cur)-1);
    dfs(ot,cur+1,cnt+1);
}
void solve14(const quantum::Printer &ot)
{
    int i, j;
    ot.Begin();
    for (C[0][0] = 1, i = 1; i <= 5; i++)
    {
        for (C[i][0] = C[i][i] = 1, j = 1; j < i; j++)
            C[i][j] = C[i - 1][j - 1] + C[i - 1][j];
        for (j = i + 1; j <= 5; j++)
            C[i][j] = 0;
    }
    dfs(ot, 0, 0);
    ot.End();
}
```

任务15

同任务14，如果需要卡常，可以先对前5个比特进行任务14的操作（全为两个1），然后对比特5进行 `H` 后控制 0、1、2、3、4的 `X`，则比特5为1时有4个1，为0时有两个1，再对比特5进行 `X` 即可。

```
void solve15(const quantum::Printer &ot)
{
    int i, j;
    ot.Begin();
    for (C[0][0] = 1, i = 1; i <= 5; i++)
    {
        for (C[i][0] = C[i][i] = 1, j = 1; j < i; j++)
            C[i][j] = C[i - 1][j - 1] + C[i - 1][j];
        for (j = i + 1; j <= 5; j++)
            C[i][j] = 0;
    }
    dfs(ot, 0, 0);
    ot.H(5);
    for(i=0;i<5;i++) ot.X(i,1<<5);
    ot.X(5);
    ot.End();
}
```

任务16

显然各用比特0、1、2控制一次比特3的`X`即可。

```
void solve16(const quantum::Printer &ot)
{
    ot.Begin();
    ot.X(3, 1), ot.X(3, 2), ot.X(3, 4);
    ot.End();
}
```

任务17

~~类似三项式反演。~~先用单个比特控制对比特4进行 `X`，发现两个是1的情况结果是对的，三个是1的情况是不对的，就再用三个比特控制对比特4进行 `X`，然后发现四个是1的情况也对了。

```
void solve17(const quantum::Printer &ot)
{
    ot.Begin();
    ot.X(4, 1), ot.X(4, 2), ot.X(4, 4), ot.X(4, 8);
    ot.X(4, 14), ot.X(4, 13), ot.X(4, 11), ot.X(4, 7);
    ot.End();
}
```

任务18

比特1是 $(|0\rangle - |1\rangle)/\sqrt{2}$，可以发现用比特0控制比特1进行 `X` 相当于对比特0为1的系数乘以 -1，那么只要先用 `H` 把比特0变成 $(|0\rangle + |1\rangle)/\sqrt{2}$，然后经过操作后变为 $(|0\rangle \pm |1\rangle)/\sqrt{2}$，再对比特0进行 `H` 就会得到 $|0\rangle$ 或 $|1\rangle$，就可以通过测量来分辨了。

```
void solve18(const quantum::Printer &ot)
{
    ot.Begin();
    ot.H(0);
    ot.End();

    ot.Begin();
    ot.H(0);
    ot.M(0);
    ot.ReturnFull({0, 1});
    ot.End();
}
```

任务19

如果能生成 $(|100\rangle + |010\rangle + |001\rangle + |111\rangle)/2$，然后再三个代码块分别测量对应的比特，则可以满足要求。生成这个状态需要对比特0、1用 `H`，然后以比特0、1为操作比特各对比特2进行 `X`，再对比特2进行 `X`，则可以生成所有奇数个1的。但是这样操作次数会超，我们可以通过不进行最后一次取反，在返回值里取反，即可避免超出限制。

```
void solve19(const quantum::Printer &ot)
{
    ot.Begin();
    ot.H(0),ot.H(1),ot.X(2,1),ot.X(2,2);
    ot.End();

    ot.Begin();
    ot.M(0),ot.ReturnFull({0,1});
    ot.End();

    ot.Begin();
    ot.M(1), ot.ReturnFull({0, 1});
    ot.End();

    ot.Begin();
```

```
    ot.M(2), ot.ReturnFull({1, 0});
    ot.End();
}
```

任务20

~~根据答案编题。~~造一个 $(|00\rangle + |11\rangle)/\sqrt{2}$，在代码块3、5中先进行 `H` 再进行测量，在代码块2、4中直接测量，计算得如果都进行 `H` 或者都不进行 `H`，则不变；否则会变成 $(|00\rangle + |10\rangle + |01\rangle - |11\rangle)/\sqrt{2}$。这样就满足要求了。

```
void solve20(const quantum::Printer &ot)
{
    ot.Begin();
    ot.H(0),ot.X(1,1);
    ot.End();

    ot.Begin();
    ot.M(0), ot.ReturnFull({0, 1});
    ot.End();

    ot.Begin();
    ot.H(0), ot.M(0), ot.ReturnFull({0, 1});
    ot.End();

    ot.Begin();
    ot.M(1), ot.ReturnFull({0, 1});
    ot.End();

    ot.Begin();
    ot.H(1), ot.M(1), ot.ReturnFull({0, 1});
    ot.End();
}
```