基础题GIT题

写在前面：

这个是我实践网址：https://github.com/Love-Suzuran/Synx-newcomer-basis

当我实践完所有的git操作后，谁懂我突然看见要写markdown的救赎感？

于是在方XX的提醒下，我**的重做重写了一遍QAQ

因为写了一遍c，~~再写一遍感觉很烦~~，所以我用了我熟练的c++写第二遍，写得快点。

然后，github里面的代码我才发现我没有打srand(time(NULL))，请见谅。

T0：

似乎首先要关联自己的github账号，直接在cmd里面这么打就好了（ps:库是public类型，可以查看滴）

```
//配置用户信息（首次使用需要）
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
//初始化Git仓库
git init
//配置远程仓库（如果有）
git remote add origin https://github.com/username/repository.git
(这个在github上面有详细的步骤记录)
```

T1：

建立分支，很简单

```
//创建并切换到分支A
git checkout -b A
```

T2:

首先打码：

```
#include<bits/stdc++.h>
using namespace std;
bool judge(int a){
    printf("电脑正在出拳\n");
    int b = rand();

    b%=3;
    b+=1;
    printf("电脑选择的是数字%d\n",b);
    if(a==b){
        printf("平局了孩子\n");
        return 0;
    }
    else if ((a==1&&b==2)||(a==2&&b==3)||(a==3&&b==1)){
```

```
        printf("你赢了孩子\n");
        return 1;
    }
    else {
        printf("你没了孩子\n");
        return 0;
    }
}
int main(){
    srand(time(NULL));
    printf("哦孩子，欢迎来到猜拳游戏。石头-1，剪刀-2，布-3\n");
    bool k =0;
    while(!k){
        printf("请输入你的选择：\n");
        int a;
        scanf("%d",&a);
        k=judge(a);
    }

    return 0;
}
```

然后更改分支A并且push到github上：

```
git add finger-guessing.cpp
git commit -m "初始版本：基础猜拳游戏，赢一次即退出"
git push origin A
```

T3：

首先改码

```
#include<bits/stdc++.h>
using namespace std;
bool judge(int a){
    printf("电脑正在出拳\n");
    int b = rand();

    b%=3;
    b+=1;
    printf("电脑选择的是数字%d\n",b);
    if(a==b){
        printf("平局了孩子\n");
        return 0;
    }
    else if ((a==1&&b==2)||(a==2&&b==3)||(a==3&&b==1)){
        printf("你赢了孩子\n");
        return 1;
    }
    else {
        printf("你没了孩子\n");
        return 0;
    }
}

int main(){
```

```
    srand(time(NULL));
    std::time_t start_time = std::time(nullptr);
    printf("哦孩子，欢迎来到猜拳游戏。石头-1，剪刀-2，布-3\n");
    bool k =0;
    while(!k){
        std::time_t end_time = std::time(nullptr);
        if(end_time - start_time>10){
            printf("已经超时10s，你GG了");
            return 0;
        }
        printf("请输入你的选择：\n");
        int a;
        scanf("%d",&a);
        k=judge(a);
    }

    return 0;
}
```

然后还是同样的做法

```
git add finger-guessing.cpp
git commit -m "添加计时功能：限时10秒获胜"
git push origin A
```

这时候去github上面查看，可以发现分支A下面的这个文件已经更新为新的版本。

同时，我们发现，通过commits可以看到以前的版本和代码。

而且我们还可以发现，在最新的代码中，详细标注了你添加的代码和删除的代码，非常好玩。

T4：

首先还是在cmd中输入：

```
git log --oneline
git reset --hard <初始版本的commit hash>
（输入的第二条语句中的hash是在第一条执行后给出的一串码，每个历史版本都会有不同的码）
git push origin A --force
```

然后呢，我们会发现我们的代码变了回去。

到github上面，也发现commits少了一条，就像是更新的版本不存在一样。

然后呢，就好像这个刚刚的更新全部白做一样，直接消失的无影无踪了。

T5：

先打代码：

```
#include<bits/stdc++.h>
using namespace std;
int ct_cpu=0;
int ct_user_win=0;
bool judge(int a){
    printf("电脑正在出拳\n");
```

```
    int b = rand();

    b%=3;
    b+=1;
    printf("电脑选择的是数字%d\n",b);
    if(a==b){
        printf("平局了孩子\n");
        printf("本次不计分\n");
        printf("现在比分你：电脑=%d:%d\n",ct_user_win,ct_cpu);
        return 0;
    }
    else if ((a==1&&b==2)||(a==2&&b==3)||(a==3&&b==1)){
        printf("你赢了孩子\n");
        printf("加分了！\n");
        ct_user_win++;
        printf("现在比分你：电脑=%d:%d\n",ct_user_win,ct_cpu);
        return 1;
    }
    else {
        printf("你没了孩子\n");
        printf("对面加分了！\n");
        ct_cpu++;
        printf("现在比分你：电脑=%d:%d\n",ct_user_win,ct_cpu);
        return 0;
    }

}
int main(){
    srand(time(NULL));
    printf("哦孩子，欢迎来到猜拳游戏。石头-1，剪刀-2，布-3\n");
    bool k =0;
    while(1){
        if(ct_cpu==3||ct_user_win==3){
            return 0;
        }
        printf("请输入你的选择：\n");
        int a;
        scanf("%d",&a);
        k=judge(a);
    }

    return 0;
}
```

然后还是同样的做法

```
git add finger-guessing.cpp
git commit -m "改为五局三胜制，平局不记录"
git push origin A
```

我们发现，更新之后的代码“改变量”是基于最初的代码的。

这也证明了上一个操作确实做得很干净，一点对后续影响也没有。

T6：

我们接着做B分支：

```
git checkout -b B
```

然后还是打码awa:

```
#include<bits/stdc++.h>
using namespace std;
int ct_cpu=0;
int ct_user_win=0;
bool judge(int a){
    printf("电脑正在出拳\n");
    int b = rand();

    b%=3;
    b+=1;
    printf("电脑选择的是数字%d\n",b);
    if(a==b){
        printf("平局了孩子\n");
        printf("本次不计分\n");
        printf("现在比分你：电脑=%d:%d\n",ct_user_win,ct_cpu);
        return 0;
    }
    else if ((a==1&&b==2)||(a==2&&b==3)||(a==3&&b==1)){
        printf("你赢了孩子\n");
        printf("加分了！\n");
        ct_user_win++;
        printf("现在比分你：电脑=%d:%d\n",ct_user_win,ct_cpu);
        return 1;
    }
    else {
        printf("你没了孩子\n");
        printf("对面加分了！\n");
        ct_cpu++;
        printf("现在比分你：电脑=%d:%d\n",ct_user_win,ct_cpu);
        return 0;
    }

}
int main(){
    printf("哦孩子，欢迎来到猜拳游戏。石头-1，剪刀-2，布-3,作弊zzz\n");
    bool k =0;
    srand(time(NULL));
    while(1){
        if(ct_cpu==3||ct_user_win==3){
            return 0;
        }
        printf("请输入你的选择：\n");
        string a;
        cin >> a;
        if(a.size()==1){
            int chose=int(a[0]);
            chose-=48;
            judge(chose);
        }
```

```
        else if(a[0]==a[1]&&a[1]==a[2]&&a[2]=='z'){
            printf("你赢了孩子\n");
            printf("加分了！\n");
            ct_user_win++;
            printf("现在比分你：电脑=%d:%d\n",ct_user_win,ct_cpu);
        }
        else{
            printf("不是哥们，你真要考察输出范围外我会怎么写吗\n");
            return 0;
        }
    }

    return 0;
}
```

然后还是老一套处理方法：

```
git add finger-guessing.cpp
git commit -m "添加作弊功能：输入zzz激活每把必赢模式"
git push origin B
```

这里我们回到github上面，发现创立了新的分支B，并且B里面本来就包含了所有A中本来含有的所有版本。

就相当于复制了一份，并且在复制的一份里面新做了一份代码。

T7：

首先要记得返回A分支：

```
git checkout A
```

打代码（只是改了跳出条件而已）

```
#include<bits/stdc++.h>
using namespace std;
int ct_cpu=0;
int ct_user_win=0;
bool judge(int a){
    printf("电脑正在出拳\n");
    int b = rand();

    b%=3;
    b+=1;
    printf("电脑选择的是数字%d\n",b);
    if(a==b){
        printf("平局了孩子\n");
        printf("本次不计分\n");
        printf("现在比分你：电脑=%d:%d\n",ct_user_win,ct_cpu);
        return 0;
    }
    else if ((a==1&&b==2)||(a==2&&b==3)||(a==3&&b==1)){
        printf("你赢了孩子\n");
        printf("加分了！\n");
        ct_user_win++;
        printf("现在比分你：电脑=%d:%d\n",ct_user_win,ct_cpu);
```

```
        return 1;
    }
    else {
        printf("你没了孩子\n");
        printf("对面加分了！\n");
        ct_cpu++;
        printf("现在比分你：电脑=%d:%d\n",ct_user_win,ct_cpu);
        return 0;
    }

}
int main(){
    srand(time(NULL));
    printf("哦孩子，欢迎来到猜拳游戏。石头-1，剪刀-2，布-3\n");
    bool k =0;
    while(1){
        if(ct_cpu==2||ct_user_win==2){
            return 0;
        }
        printf("请输入你的选择：\n");
        int a;
        scanf("%d",&a);
        k=judge(a);
    }

    return 0;
}
```

然后还是老方法一套走：

```
git add finger-guessing.c
git commit -m "改为三局两胜制"
git push origin A
```

我们发现，A分支中并没有B中刚刚新建立的代码，所以这一次爆出的修改还是根据五局三胜的代码改过来的。

T8：

这个操作我觉得是最nb的

```

git checkout A
git merge B
git push origin A
```

这里我们就将B的最新代码和A的最新代码进行了融合

有一种求并集的感觉。

发现两个合起来同时满足了两个条件

真的很不错

（但我觉得可能很有限制，不是想融合就融合，而且可能会有很大的误差）

最后给出他自动合并后的代码：

```cpp
#include<bits/stdc++.h>
using namespace std;
int ct_cpu=0;
int ct_user_win=0;
bool judge(int a){
    printf("电脑正在出拳\n");
    int b = rand();
    b%=3;
    b+=1;
    printf("电脑选择的是数字%d\n",b);
    if(a==b){
        printf("平局了孩子\n");
        printf("本次不计分\n");
        printf("现在比分你：电脑=%d:%d\n",ct_user_win,ct_cpu);
        return 0;
    }
    else if ((a==1&&b==2)||(a==2&&b==3)||(a==3&&b==1)){
        printf("你赢了孩子\n");
        printf("加分了！\n");
        ct_user_win++;
        printf("现在比分你：电脑=%d:%d\n",ct_user_win,ct_cpu);
        return 1;
    }
    else {
        printf("你没了孩子\n");
        printf("对面加分了！\n");
        ct_cpu++;
        printf("现在比分你：电脑=%d:%d\n",ct_user_win,ct_cpu);
        return 0;
    }

}
int main(){
    printf("哦孩子，欢迎来到猜拳游戏。石头-1，剪刀-2，布-3,作弊zzz\n");
    bool k =0;
    while(1){
        if(ct_cpu==2||ct_user_win==2){
            return 0;
        }
        printf("请输入你的选择：\n");
        string a;
        cin >> a;
        if(a.size()==1){
            int chose=int(a[0]);
            chose-=48;
            judge(chose);
        }
        else if(a[0]==a[1]&&a[1]==a[2]&&a[2]=='z'){
            printf("你赢了孩子\n");
            printf("加分了！\n");
            ct_user_win++;
            printf("现在比分你：电脑=%d:%d\n",ct_user_win,ct_cpu);
        }
        else{
            printf("不是哥们，你真要考察输出范围外我会怎么写吗\n");
            return 0;
        }
```

```
    }

    return 0;
}
```