# 고급프로그래밍및실습 과제 #6 (12주차)

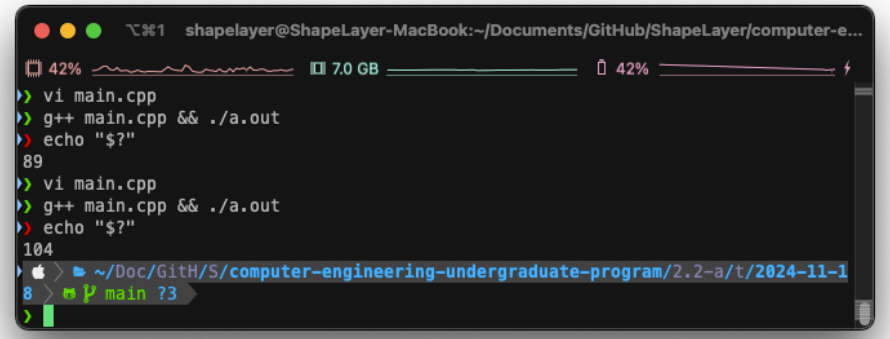
214823 박종현

1. `static_cast`
2. `dynamic_cast`
3. `const_cast`
4. `reinterpret_cast`
5. (선택) 다중 예외처리

## 답안의 의도한 실행 결과

- 표준 출력 없음
- 호스트에게 `0` 리턴

발생해서는 안되는 상황




- 의도적으로 예외를 발생시키는 코드에서 프로세스 호스트에게 다음 오류 코드를 리턴
  - `ECANCELED(89)` : 예외 발생 중 `AssertionFailedException`이 아닌 `exception` 발생
  - `ENOTRECOVERABLE(104)` : 아무 오류도 발생하지 않음

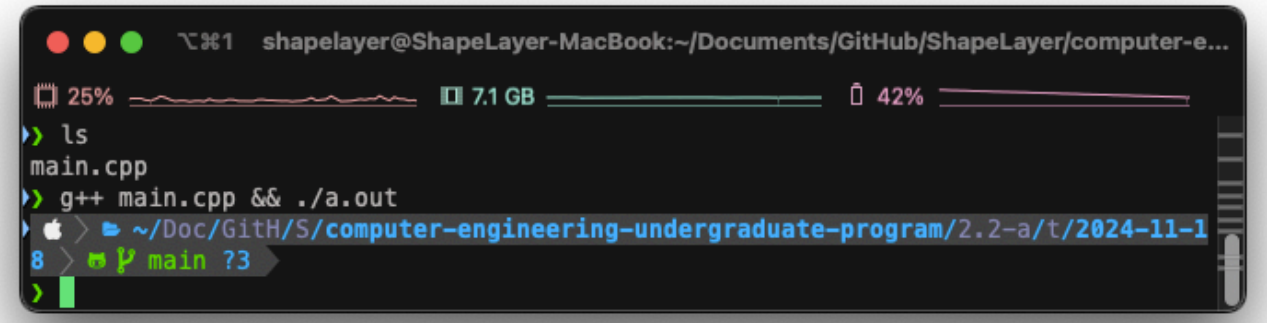
## 답안

```cpp
#include "bits/stdc++.h"
#define UUID_SIZE 37
#define _ASSERT(expr) do { if (!(expr)) throw AssertionFailedException(); } while (0)

enum ResponseState {
  ok,
  error,
};

class CommonResponse {
private:
  ResponseState state;
public:
  CommonResponse(ResponseState state): state(state) {}
  ResponseState getState() const {
    return state;
  }
};

class AssertionFailedException : public exception {};
```

```
21
22  class UUID {
23  private:
24    uint32_t time_low;
25    uint16_t time_mid;
26    uint16_t time_hi_and_version;
27    uint16_t clock_seq;
28    uint64_t node;
29  public:
30    UUID(uint32_t time_low, uint16_t time_mid, uint16_t time_hi_and_version, uint16_t clock_seq, uint64_t node): time_low(time_low), time_mid(time_mid), time_hi_and_version(time_hi_and_version), clock_seq(clock_seq), node(node) {}
31    void get(char* buffer) const {
32      snprintf(buffer, sizeof(buffer), "%08x-%04x-%04x-%04x-%012llx",
33        time_low,
34        time_mid,
35        time_hi_and_version,
36        clock_seq,
37        node);
38    }
39    void set(char* buffer) {
40      if (!buffer || strlen(buffer) != 36) {
41        return;
42      }
43
44      unsigned int tl;
45      unsigned short tm, thv, cs;
46      unsigned long long n;
47
48      if (sscanf(buffer, "%08x-%04hx-%04hx-%04hx-%012llx",
49                 &tl, &tm, &thv, &cs, &n) == 5) {
50        time_low = tl;
51        time_mid = tm;
52        time_hi_and_version = thv;
53        clock_seq = cs;
54        node = n;
55      }
56    }
57  };
58
59  class AuthorizationResponse : public CommonResponse {
60  private:
61    UUID secret_key;
62  public:
63    AuthorizationResponse(ResponseState state, UUID secret_key): CommonResponse(state), secret_key(secret_key) {}
64    UUID get_secret_key() const {
65      return secret_key;
66    }
67  };
68
69  int main() {
70    CommonResponse cr = CommonResponse(ok);
71    AuthorizationResponse ar = AuthorizationResponse(ok, UUID(0, 0, 0, 0, 0));
72
73    CommonResponse ar__cr_converted = *dynamic_cast<CommonResponse*>(&ar);
```

```
  AuthorizationResponse cr__ar_converted = *static_cast<AuthorizationResponse*>(&cr);

  _ASSERT(typeid(cr) == typeid(CommonResponse));
  _ASSERT(typeid(ar) == typeid(AuthorizationResponse));
  _ASSERT(typeid(ar__cr_converted) == typeid(CommonResponse));
  _ASSERT(typeid(cr__ar_converted) == typeid(AuthorizationResponse));


  const UUID const_uuid = UUID(0x12345678, 0x9ABC, 0xDEF0, 0x1234, 0x567890ABCDEF);

  char ar_uuid_buf[UUID_SIZE];
  ar.get_secret_key().get(ar_uuid_buf);

  UUID const_uuid_new = (*const_cast<UUID*>(&const_uuid));
  const_uuid_new.set(ar_uuid_buf);

  char const_uuid_buf[UUID_SIZE];
  const_uuid_new.get(const_uuid_buf);

  _ASSERT(strcmp(ar_uuid_buf, const_uuid_buf));


  CommonResponse *pcr = &cr;
  AuthorizationResponse *par = &ar;

  cr__ar_converted = *(reinterpret_cast<AuthorizationResponse*>(pcr));
  ar__cr_converted = *(reinterpret_cast<CommonResponse*>(par));
  _ASSERT(typeid(cr__ar_converted) == typeid(ar));
  _ASSERT(typeid(ar__cr_converted) == typeid(cr));

  try {
    _ASSERT(typeid(cr__ar_converted) == typeid(cr));
  } catch (const AssertionFailedException e) {
    return 0;
  } catch (const exception e) {
    return ECANCELED;
  }
  return ENOTRECOVERABLE;
}

```

## 실행 결과


⌥⌘1 shapelayer@ShapeLayer-MacBook:~/Documents/GitHub/ShapeLayer/computer-e...
25%
7.1 GB
42%
ls
main.cpp
g++ main.cpp && ./a.out
~/Doc/GitH/S/computer-engineering-undergraduate-program/2.2-a/t/2024-11-18
main ?3