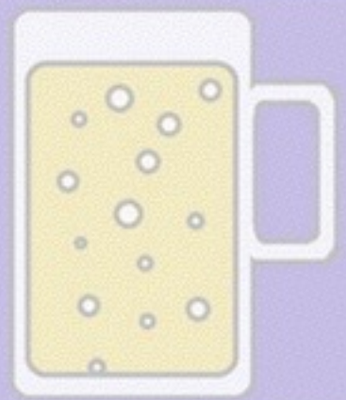
#BeeRumP

Des rumps et de la bière



HE'S THE MEANEST SON OF A SNAKE YOU'VE EVER SEEN!


#BeeRumP
Des rumps et de la bière
Haskell
HE'S THE ...
HKM
KILLING
MACHINE
U.S. GOLD
CARVE OUT A PATH OF SAVAGE DESTRUCTION AS YOU BATTLE TO STAMP YOUR SUPREMACY OVER A MULTITUDE OF OPPONENTS.
Face Igor the Fearless and his rabid dog amongst the sacred Temples of Moscow. Don't be fooled by ladies of the night, Maria and Helga, streetwise and toughened in the seedy underworld of Amsterdam. Match the cunning guile of Miguel, master bullfighter and the unvanquished fighting bull Brutus in the splendid surroundings of the Barcelona bull ring. Trade blows with the awesome titan Hans and his drunken compatriot outside a German beerhouse. Finally confront the merciless terrorists of the Middle East amongst the battle torn ruins of Beirut.
Tough and mean .... you're the Human Killing Machine!
U.S. Gold Ltd., Units 2/3 Holford Way, Holford, Birmingham B6 7AX.

“the awesome titan HANS”

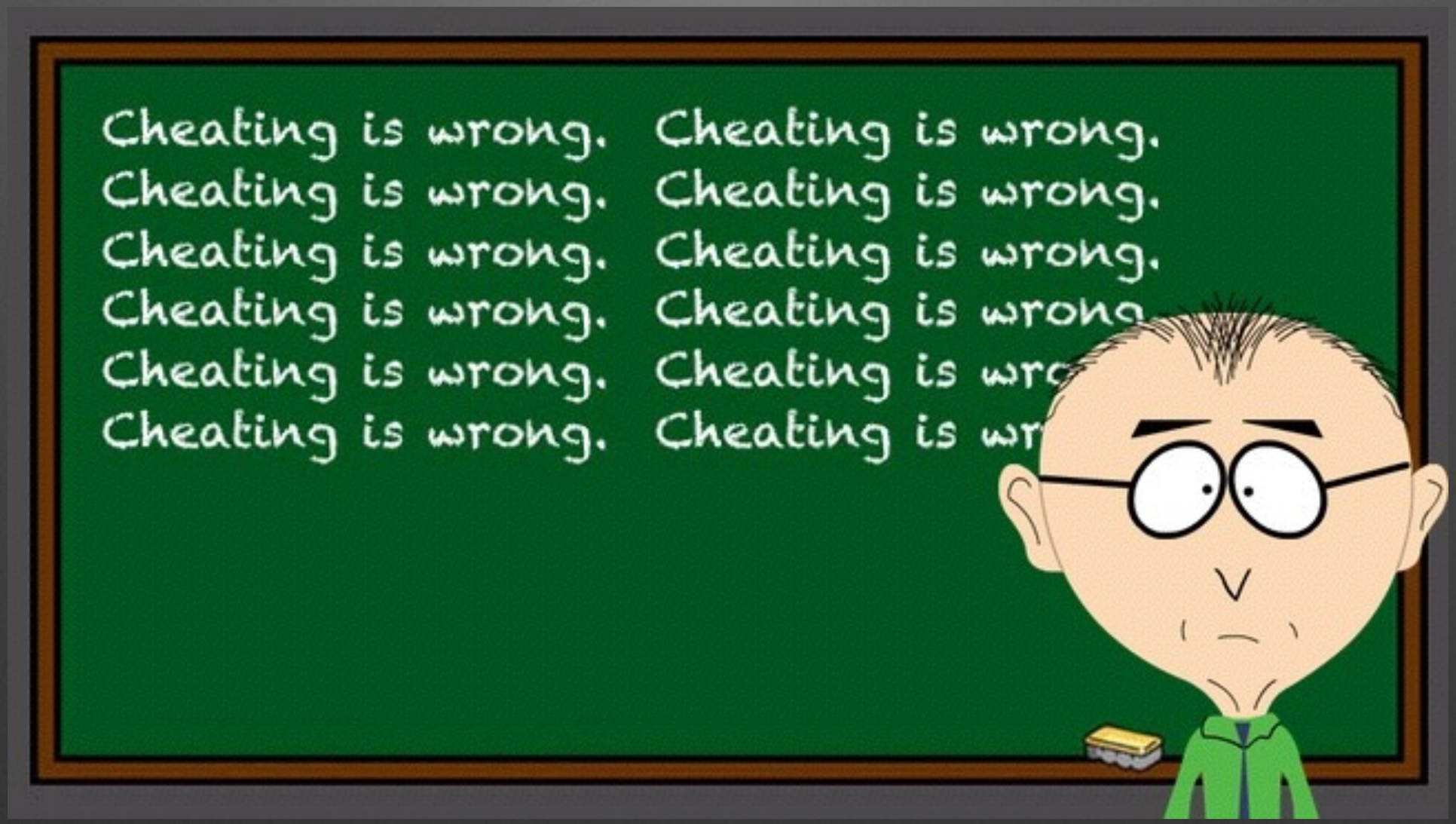
Tough Guy, 1995

Game Trainers

Cheats…


BE A COUNTER-STRIKE GOD
FOR ONLY
$10.95
PER MONTH!
See through walls!
Use an aimbot to
score killer headshots!
You Won't BELIEVE these hacks!

…are bad, m’kay?


Cheating is wrong. Cheating is wrong.
Cheating is wrong. Cheating is wrong.
Cheating is wrong. Cheating is wrong.
Cheating is wrong. Cheating is wrong
Cheating is wrong. Cheating is wr
Cheating is wrong. Cheating is wr

PEEK and POKE

```
// Read world pointer
if (!ReadProcessMemory(hProc, moduleBase + WORLD_OFFSET,
                       &worldOff, sizeof(worldOff), &read)) {
    return FALSE;
}
if (read != sizeof(worldOff)) {
    return FALSE;
}
if (!isValidPtr(worldOff)) {
    return FALSE;
}
```

```
// Read world pointer
READ_VALID_POINTER(hProc, moduleBase + WORLD_OFFSET, &worldOff,
                   return FALSE);
// Read player pointer
READ_VALID_POINTER(hProc, worldOff + PLAYER_OFFSET, &playerOff,
                   return FALSE);
```

Haskell

- Purely functional
- Statically typed
- Lazy

```
primes = filterPrime [2..]
  where filterPrime (p:xs) =
          p : filterPrime [x | x <- xs, x `mod` p /= 0]
```

Haskell

- Category theory
- Lambda calculus
- Combinatory logic

XKCD #1312

Maybe type

```haskell
data Maybe a = Just a | Nothing

readPointer :: Addr -> Maybe Addr
readPointer _ = Nothing --TODO :(
```

```
PVOID readPointer(LPCVOID address) {
    return NULL;
}
```

Monad

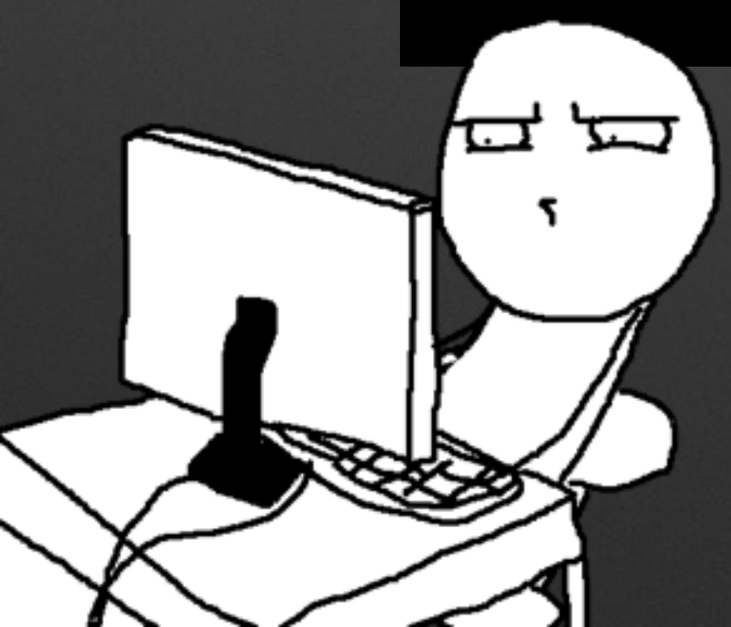
“a monad is a **purely abstract** concept, with no fundamental relationship to **anything** you've probably ever heard of before”

“if it looks like a monad, and acts like a monad, it is a monad”

Mike Vanier, 2010

Maybe Monad

```
instance Monad Maybe where
    (Just x)  >>= k       = k x
    (Just _)  >>  k       = k
    Nothing   >>= _       = Nothing
    return                = Just
```



Maybe Monad

```
readProcessMemory :: ProcessHandle -> Addr -> Int -> Maybe Bytes
readProcessMemory _ _ _ = Just [0xDE, 0xAD] -- FIXME


readSomeBytes :: ProcessHandle -> Addr -> Maybe Bytes
readSomeBytes proc addr = do
    fourBytes <- readProcessMemory proc addr 4
    twoMoreBytes <- readProcessMemory proc (addr + 10) 2
    guard $ head fourBytes == 0xDE
    return $ fourBytes ++ twoMoreBytes
```

FFI

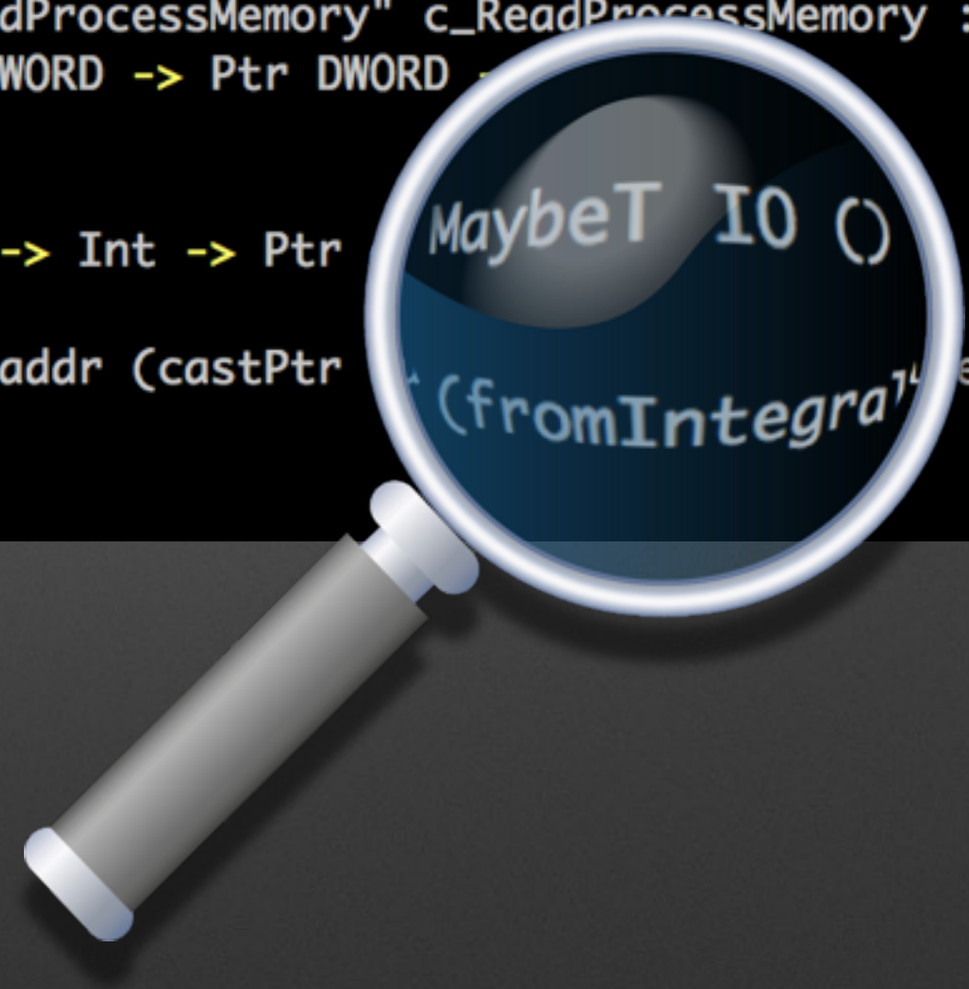
```haskell
foreign import WINDOWS_CCONV "windows.h ReadProcessMemory" c_ReadProcessMemory ::
    ProcessHandle -> Addr -> Ptr Word8 -> DWORD -> Ptr DWORD -> IO BOOL


peekProcessMemory :: ProcessHandle -> Addr -> Int -> Ptr a -> MaybeT IO ()
peekProcessMemory proc addr size buf = do
    res <- lift $ c_ReadProcessMemory proc addr (castPtr buf) (fromIntegral size) nullPtr
    guard res
    return ()
```

FFI

```haskell
foreign import WINDOWS_CCONV "windows.h ReadProcessMemory" c_ReadProcessMemory ::
    ProcessHandle -> Addr -> Ptr Word8 -> DWORD -> Ptr DWORD -


peekProcessMemory :: ProcessHandle -> Addr -> Int -> Ptr
peekProcessMemory proc addr size buf = do
    res <- lift $ c_ReadProcessMemory proc addr (castPtr  (fromIntegra  e) nullPtr
    guard res
    return ()
```



Monad Transformers

MTL (Monad Transformer Library)

- 標準のモナドライブラリ
  - Preludeのモナドを大幅強化
- これらのものを含む
  - 幾つかの標準的なモナド
  - これらのモナドを合成するための
    モナド変換子(Monad Transformers)

Real World Haskell

```
getWorld :: ProcessHandle -> Addr -> MaybeT IO Addr
getWorld proc base = do
    gWorld <- readPtr32 proc $ base *+ gWorldOffset
    onDebug . putStrLn $ "World: " ++ show gWorld
    return gWorld
```

Real World Haskell

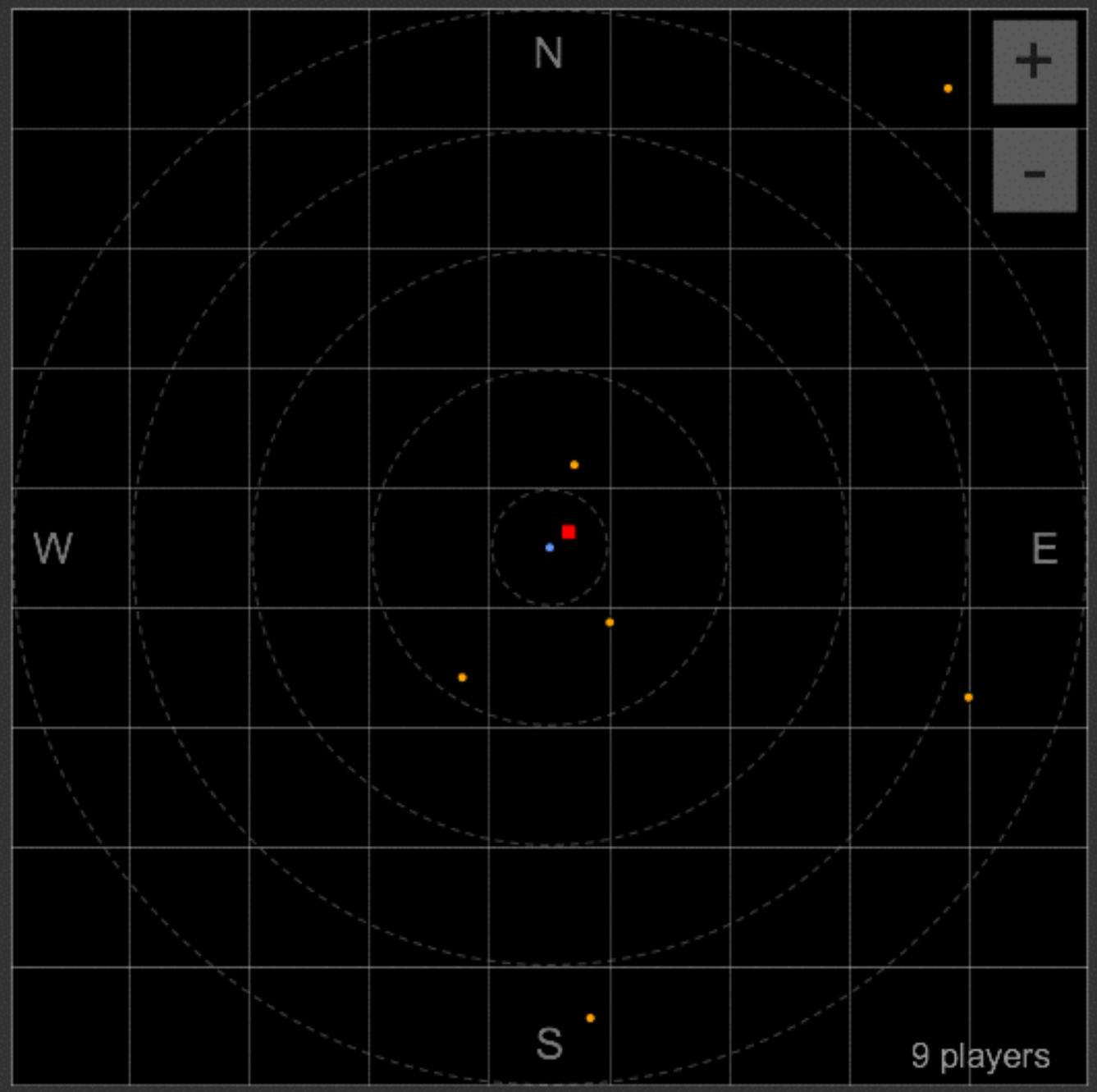
```
readRString :: ProcessHandle -> Addr -> MaybeT IO String
readRString proc addr = do
    strLen <- readUInt32 proc $ addr *+ 4
    guard (strLen < 256) -- arbitrary limit strings to 256 chars
    str <- readProcessMemory proc (addr *+ 8) . fromIntegral $ strLen - 1
    return $ BS.unpack str
```

```
readEntityType :: ProcessHandle -> Addr -> MaybeT IO String
readEntityType proc entity = do
    typPtr <- readValidPtr32 proc $ entity *+ gEntityTypePtrOffset
    typStrPtr <- readValidPtr32 proc $ typPtr *+ gTypeStrOffset
    readRString proc typStrPtr
```

Real World Haskell

```haskell
readGameData :: ProcessHandle -> Addr -> IO (Maybe GameData)
readGameData proc base = runMaybeT $ do
    -- read world pointer
    gWorld <- getWorld proc base
    -- read network manager pointer
    gNetMgr <- getNetMgr proc base
    -- read player identities
    pids <- liftMaybe . getPlayersIds proc $ gNetMgr
    -- read current player data
    player <- getPlayer proc gWorld pids
    -- read all players data
    players <- getPlayers proc gWorld pids
    -- GameData
    return $ GameData player players
```

```haskell
main :: IO ()
main = do
    -- Find process id
    pid_ <- processNameToPid gAppName
    pid <- assert "app not found" pid_
    -- Find remote module
    mHdl_ <- getRemoteModuleHandle pid gAppName
    mHdl <- assert "main module not found" mHdl_
    -- Start web server
    scotty gServerPort $ do
        -- handle index
        get "/" $ html indexHtml
        -- handle "/data" requests
        get "/data" $ do
            -- Connect to process
            -- and read game data
            gameData <- liftIO $ readGameData pid mHdl
            case gameData of
                Just g -> json g
                Nothing -> serverError "unable to retrieve game data"
        -- unhandled requests
        notFound $ serverError "invalid path"
```



Questions?

More about Haskell

- https://www.haskell.org
- http://learnyouahaskell.com
- http://dev.stephendiehl.com/hask/