1. (10 points) Give the output for the following program.

```cpp
1   #include <iostream>
2   class Game {
3   public:
4     Game()            { std::cout << "default" << std::endl;    }
5     Game(const char*) { std::cout << "convert" << std::endl;    }
6     Game(const Game&) { std::cout << "copy" << std::endl;       }
7     ~Game()           { std::cout << "destructor" << std::endl; }
8     Game& operator=(const Game&) {
9       std::cout << "copy assign" << std::endl;
10      return *this;
11    }
12  };
13  Game fun(Game g) {
14    return g;
15  }
16
17  int main() {
18    Game cat("cat");
19    fun(cat);
20  }
```

```
convert
copy
copy
destructor
destructor
destructor
```

---

2. (10 points) Give the output for the following program.

```cpp
1   #include <iostream>
2   class Game {
3   public:
4     Game()            { std::cout << "default" << std::endl;    }
5     Game(const char*) { std::cout << "convert" << std::endl;    }
6     Game(const Game&) { std::cout << "copy" << std::endl;       }
7     ~Game()           { std::cout << "destructor" << std::endl; }
8     Game& operator=(const Game&) {
9       std::cout << "copy assign" << std::endl;
10      return *this;
11    }
12  };
13  Game fun() {
14    return Game( Game() );
15  }
16  int main() {
17    Game cat("cat");
18    fun();
19  }
```

```
convert
default
destructor
destructor
```

1

3. (10 points) Give the output for the following program.

```cpp
#include <iostream>

void f(const int& x)  { std::cout << "l-value ref: " << x << std::endl; }
void f(int&& x)       { std::cout << "r-value ref: " << x << std::endl; }
int  f()              { return 19; }

int main() {
   f(20);
   int x = 7;
   f(x);
   f(std::move(x));
   f(x+1);
   f(f());
}
```

```
r-value ref: 20
l-value ref: 7
r-value ref: 7
r-value ref: 8
r-value ref: 19
```

4. (10 points) Give the output for the following program.

```cpp
#include <iostream>

class A{
public:
  A()           { std::cout << "default constructor" << std::endl; }
  A(int)        { std::cout << "conversion constructor" << std::endl; }
  A(const A&)   { std::cout << "copy constructor" << std::endl; }
  A(const A&&)  { std::cout << "move constructor" << std::endl; }
  A& operator=(const A&) {
     std::cout << "copy assignment" << std::endl;
     return *this;
  }
  A& operator=(const A&&) {
     std::cout << "move assignment" << std::endl;
     return *this;
  }
};

int main() {
  A a, b = a;
  a = b;
  b = 99;
}
```

```
default constructor
copy constructor
copy assignment
conversion constructor
move assignment
```

2

5. (10 points) Give the output for the following program.

```cpp
1   #include <iostream>
2   #include <vector>
3   class Game {
4   public:
5     Game()             { std::cout << "default" << std::endl;    }
6     Game(const char*)  { std::cout << "convert" << std::endl;    }
7     Game(const Game&)  { std::cout << "copy" << std::endl;       }
8     ~Game()            { std::cout << "destructor" << std::endl; }
9   private:
10    const char* name;
11  };
12  Game fun(Game g) {
13    return g;
14  }
15
16  int main() {
17    std::vector<Game> games;
18    games.emplace_back("Monopoly");
19    games.push_back("Magic the Gathering");
20  }
```

```
convert
convert
copy
copy
destructor
destructor
destructor
destructor
```

---

6. (10 points) Write a move constructor for class Game above.

```cpp
Game(Game&& g) : name(std::move(g.name)) {
  g.name = nullptr;
}
```

or

```cpp
Game(Game&& g) : name(std::exchange(g.name, nullptr)) {}
```

3

7. (15 points) Give the output for the following program.

```cpp
1   #include <iostream>
2   #include <string>
3   class A {
4   public:
5     A(int n) : number(n) {}
6     ~A() { std::cout << "deleting A" << std::endl; }
7     int getNumber() const { return number; }
8     virtual std::string getName() const { return "I'm A"; }
9   private:
10    int number;
11  };
12  class B : public A {
13  public:
14    B(int m, int n) : A(m), number(n) {}
15    ~B() { std::cout << "deleting B" << std::endl; }
16    int getNumber() const { return number; }
17    virtual std::string getName() const { return "I'm B"; }
18  private:
19    int number;
20  };
21  int main() {
22    A* x = new B(12, 13);
23    B* y = new B(66, 77);
24    A* z = new B(1, 2);
25    std::cout << x->getNumber() << std::endl;
26    std::cout << x->getName() << std::endl;
27    std::cout << y->getNumber() << std::endl;
28    std::cout << y->getName() << std::endl;
29    delete z;
30  }
```

```
12
I'm B
77
I'm B
deleting A
```

8. (10 points) Convert the following program so that Clock is a *GoF* singleton.

```cpp
1   #include <iostream>
2
3   class Clock {
4   public:
5     static Clock* getInstance() {
6       if ( !instance ) instance = new Clock;
7       return instance;
8     }
9     int getTicks() const { return ticks; }
10    void update() { ++ticks; }
11  private:
12    int ticks;
13    static Clock* instance;
14    Clock() : ticks(0) {}
15    Clock(const Clock&);
16    Clock&  operator=(const Clock&);
17  };
18
19  Clock* Clock::instance = NULL;
20  int main( ) {
21    Clock* clock = Clock::getInstance();
22    clock->update();
23    std::cout << clock->getTicks() << std::endl;
24  }
```

9. (15 points) The following program draws a rectangle on the screen. Modify the program so that it draws a triangle on the screen.

```cpp
1  #include <iostream>
2  #include <string>
3  #include <SDL2/SDL.h>
4
5  const int WIDTH = 640;
6  const int HEIGHT = 480;
7  const std::string TITLE = "Drawing a Rectangle";
8
9  int main (int , char *[]) {
10    SDL_Window* window = SDL_CreateWindow(
11        TITLE.c_str(),
12        SDL_WINDOWPOS_CENTERED,
13        SDL_WINDOWPOS_CENTERED,
14        WIDTH,
15        HEIGHT,
16        SDL_WINDOW_SHOWN
17    );
18
19    SDL_Renderer* renderer = SDL_CreateRenderer(
20      window, -1, SDL_RENDERER_ACCELERATED
21    );
22
23    SDL_SetRenderDrawColor( renderer, 208, 209, 210, 255 );
24    SDL_RenderClear( renderer );
25
26    SDL_Rect r = {150, 150, 250, 150};
27    SDL_SetRenderDrawColor( renderer, 255, 255, 255, 255 );
28    SDL_RenderDrawRect( renderer, &r );
29    SDL_RenderPresent(renderer);
30
31    SDL_Event event;
32    const Uint8* keystate;
33    while ( true ) {
34      keystate = SDL_GetKeyboardState(0);
35      if (keystate[SDL_SCANCODE_ESCAPE]) { break; }
36      if (SDL_PollEvent(&event)) {
37        if (event.type == SDL_QUIT) {
38          break;
39        }
40      }
41    }
42    SDL_DestroyRenderer(renderer);
43    SDL_DestroyWindow(window);
44    SDL_Quit();
45    return EXIT_SUCCESS;
46  }
```