1. (7 points) Give the output for the following program:

```
#include <iostream>
class Bird {
public:
  Bird(int w) : wingSpan(w), speed(2*wingSpan) {
    std::cout << "Speed: " << speed << std::endl;
    std::cout << "Wing Span: " << wingSpan << std::endl;
  }
private:
  int speed;
  int wingSpan;
};

int main() {
  Bird robin(5);
}
```

2. (8 points) Give the output for the following program:

```
#include <iostream>
class Number {
public:
  Number()              { std::cout << "default" << std::endl;    }
  Number(float)         { std::cout << "convert" << std::endl;    }
  Number(const Number&) { std::cout << "copy" << std::endl;       }
  ~Number()             { std::cout << "destructor" << std::endl; }
  Number& operator=(const Number&) {
    std::cout << "assign" << std::endl;
    return *this;
  }
};

class Student {
public:
  Student(float g) {
    gpa = g;
  }
private:
  Number gpa;
};

int main() {
  Student* npc = new Student(3.4);
}
```

3. (7 points) Give the output for the following program:

```
#include <iostream>

int main() {
  int number = 17;
  int* ptr = &number;
  int& ref = number;
  ref = 99;
  std::cout << *ptr  << std::endl;
  std::cout << number << std::endl;
}
```

4. (8 points) Give the output of this program.

```
#include <cstring>
#include <iostream>
class string {
public:
   string(const char* s) : buf(new char[strlen(s)+1]) { strcpy(buf, s); }
   char* getBuf() const { return buf; }
private:
   char * buf;
};
std::ostream& operator<<(std::ostream& output, const string& s) {
  return output << s.getBuf();
}

int main() {
  string a("cat"), b = a;
  char* dummy = a.getBuf();
  dummy[0] = 'r';
  std::cout << a << std::endl;
  std::cout << b << std::endl;
}
```

5. (5 points) We wrote two member functions to overload the [] operator. Write the one, not both, that is required to compile the following program:

```
#include <iostream>
#include <cstring>
class string {
public:
  string(const char* b) : buf(new char[strlen(b)+1]) {
    strcpy(buf, b);
  }
  ~string() { delete [] buf; }
private:
  char* buf;
};
int main() {
  const string a("dog");
  std::cout << a[0] << std::endl;
}
```

6. (5 points) This program crashes with a double free error. Fix the problem by adding code; you may not delete code.

```
#include <cstring>
#include <iostream>
class string {
public:
   string(const char* s) : buf(new char[strlen(s)+1]) { strcpy(buf, s); }
   ~string() { delete [] buf; }
private:
   char * buf;
};

int main() {
  string a("cat"), b = a;
}
```

7. (10 points) Answer the following two questions:

   (a) Give the output for the following program;

   (b) The prograam leaks memory. Add code to remove leaks.

```
#include <iostream>
class Number {
public:
  Number()                { std::cout << ”default” << std::endl;     }
  Number(int)             { std::cout << ”convert” << std::endl;     }
  Number(const Number&)   { std::cout << ”copy” << std::endl;        }
  ~Number()               { std::cout << ”destructor” << std::endl;  }
  Number& operator=(const Number&) {
    std::cout << ”assign” << std::endl;
    return *this;
  }
};

void f(Number n) {}

int main() {
  Number a = 17, b = a;
  f(a);
  Number * number = new Number(3);
}
```

8. (10 points) Write two functions for Student: (a) an assignment operator, and (b) Student::setMajor.

```
#include <iostream>
#include <cstring>
#include <vector>

class Student {
public:
  Student() : major(new char[1]) { major[0] = '\0'; }
  Student(const char* m) : major(new char[strlen(m)+1]) {
    strcpy(major, m);
  }
  virtual ~Student() { delete [] major; }
private:
  char* major;
};
```

9. (10 points) Give the output for the following program.

```cpp
#include <iostream>
#include <vector>
const int MAX = 3;
class Number {
public:
  Number() : number(0) { std::cout << "default" << std::endl; }
  explicit Number(int n) : number(n) {
    std::cout << "convert: " << n << std::endl;
  }
  Number(const Number& a) : number(a.number) {
    std::cout << "copy: " << a.number << std::endl;
  }
  Number& operator=(const Number& rhs) {
    if ( this != &rhs ) { number = rhs.number; }
    std::cout << "assign" << std::endl;
    return *this;
  }
  int getNumber() const { return number; }
private:
  int number;
};

void print(const std::vector<Number> & vec) {
  for (unsigned int i = 0; i < vec.size(); ++i) {
    std::cout << vec[i].getNumber()  << ", ";
  }
  std::cout << std::endl;
}

void init(std::vector<Number> & vec) {
  for (unsigned int i = 0; i < MAX; ++i) {
    vec.push_back( Number(i+1) );
  }
}

int main() {
  std::vector<Number> vec;
  vec.reserve(3);
  init(vec);
  vec.push_back( Number(4) );
  std::cout << "SIZE: " << vec.size() <<  std::endl;
  std::cout << "CAP:  " << vec.capacity() <<  std::endl;
  print(vec);
}
```

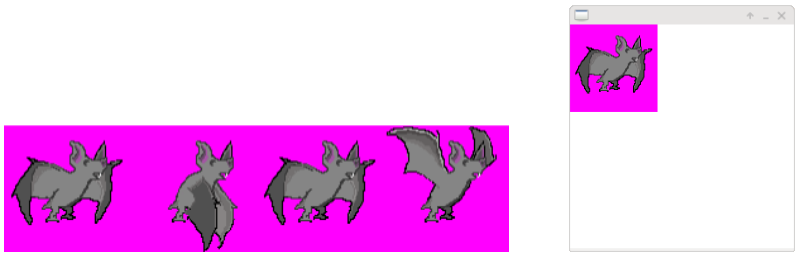
(a) Sprite sheet. (b) Single image.

Figure 1: This figure illustrates extraction of a single image from a sprite sheet.

10. (10 points) Class ExtractSurface, listed below on lines 1–7, can extract a surface from a sprite sheet, shown in Figure 1a. Convert class ExtractSurface to a GoF Singleton. Meyer's Item #6 recommends: "Explicitly disallow compiler generated functions you do not want." Be sure to disallow these functions in your GoF singleton.

```
 1 class ExtractSurface {
 2 public:
 3   SDL_Surface* get(SDL_Surface*, int, int, int, int) const;
 4 private:
 5   Uint32 getpixel(SDL_Surface*, int, int) const;
 6   void putpixel(SDL_Surface *, int, int, Uint32) const;
 7 };
 8
 9 SDL_Surface* ExtractSurface::get(SDL_Surface* source, int frameWidth,
10   int frameHeight, int topX, int topY) const {
11   SDL_Surface * croppedSurface = SDL_CreateRGBSurface(
12     SDL_SWSURFACE | SDL_SRCALPHA, frameWidth, frameHeight,
13                 source->format->BitsPerPixel, source->format->Rmask,
14                 source->format->Gmask, source->format->Bmask, source->format->Amask
15   );
16   SDL_LockSurface(croppedSurface);
17   SDL_LockSurface(source);
18   int targetX = 0;
19   int targetY = 0;
20   for (int x = topX; x < topX+frameWidth; ++x) {
21     for (int y = topY; y < frameHeight+topY; ++y) {
22       putpixel(croppedSurface, targetX, targetY, getpixel(source, x, y));
23       targetY++;
24     }
25     targetY=0; ++targetX;
26   }
27   SDL_UnlockSurface(croppedSurface); SDL_UnlockSurface(source);
28   return croppedSurface;
29 }
30
31 // Code elided from the following two methods to reduce cognitive burden
32 Uint32 ExtractSurface::getpixel(SDL_Surface *surface, int x, int y) const {}
33 void ExtractSurface::putpixel(SDL_Surface *surface, int x, int y, Uint32 p) const {}
```

11. (10 points) The following program uses non-singleton class ExtractSurface to extract a single image and blit it onto a screen, shown in Figure 1b. (a) Fix the program so that it uses the GoF singleton from the previous question; make sure your program has no compile or link errors. (b) function main has a subtle memory leaks. Add code to remove them.

```
#include <string>
#include "extractSurface.h"
const int WIDTH = 256;
const int HEIGHT = 256;

SDL_Surface* getImage(const std::string& filename, bool setColorKey) {
  SDL_Surface *temp = SDL_LoadBMP(filename.c_str());
  if (temp == NULL) {
    throw std::string("Unable to load bitmap.")+SDL_GetError();
  }
  if ( setColorKey ) {
    Uint32 colorkey = SDL_MapRGB(temp->format, 255, 0, 255);
    SDL_SetColorKey(temp, SDL_SRCCOLORKEY|SDL_RLEACCEL, colorkey);
  }
  SDL_Surface *image = SDL_DisplayFormat(temp);
  SDL_FreeSurface(temp);
  return image;
}

void drawBackground(SDL_Surface* screen) {
  SDL_FillRect( screen, NULL,
    SDL_MapRGB(screen->format, 255, 255, 255) );
  SDL_Rect dest = {0, 0, 0, 0};
  SDL_BlitSurface( screen, NULL, screen, &dest );
}

void blit(SDL_Surface* image, SDL_Surface* screen) {
  SDL_Rect src = { 0, 0, image->w, image->h };
  SDL_Rect dest = {0, 0, 0, 0 };
  SDL_BlitSurface(image, &src, screen, &dest);
}

int main(int, char*[]) {
  SDL_Init(SDL_INIT_VIDEO);
  atexit(SDL_Quit);
  SDL_Surface *screen = SDL_SetVideoMode(WIDTH, HEIGHT, 0, SDL_DOUBLEBUF);
  SDL_Surface *temp = getImage("images/bats.bmp", true);
  SDL_Surface *image = ExtractSurface().get(temp, 100, 100, 0, 0);

  drawBackground(screen);
  blit(image, screen);
  SDL_Flip(screen);
  SDL_Delay(1000);
  SDL_FreeSurface(image);
}
```

12. Give short answers to the following questions:

    (a) In Item #1, Meyer's describes C^{++} as composed of 4 sublanguages, one of these is the C language. Name 2 of the 3 remaining sublanguages.

    (b) In Item #3, Meyers gives 2 reasons for using const member functions. List these 2 reasons.

    (c) Meyer's states that the following code is illegal: (a*b) = c. Why is this important?

    (d) In Item #4 Meyer's states that for built-in types there is no difference between assignment and initialization lists. When is an initialization list more efficient than assignment and what is this gain in efficiency?

    (e) In Items #22 and #23, Meyer's explains that an object's encapsulation is inversely proportional to the amount of code that might be broken if that object changes. What should be conclude about protected data attributes of a class?