**1. Załóżmy że klasa String posiada następujące składowe prywatne:**

*class String*

*{*

*private:*

*char \* str;*

*int len;*

*...*

*}*

**a) Czy dla takiej klasy poporawny jest poniższy konstruktor:**

*String::String() {}*

**b) Czy dla takiej klasy poporawny jest poniższy konstruktor:**

*String::String(const char \* s)*

*{*

*str = s;*

*len = strlen(s);*

*}*

**c) Czy dla takiej klasy poprawny jest poniższy konstruktor:**

*String::String(const char \*)*

*{*

*strcpy(str, s);*

*len = strlen(s);*

*}*

a – może bo nic nie przypisze, Mimo to konstuktor powinien przypisać wartośc NULL skladwoje str, albo zainicjalizować go za pomocą operatora new[]

b) nie, poniewaz usuniecie str usunie spowoduje wycieki pamieci -   
len = strlen(s);

str = new char[len +1] – przydzielamy nowa pamiec do str

strccpy(str, s) kopiowanie

Ten Konstruktor nie tworzy łańcucha a jedynie kopiuje adres starego.

c- najpierw sprawdzamy dlugosc potem przypisujemy pamiec a na koncu kopiujemy – jesli zrobimy to tak jak tutaj to dojdzie do kopiowanie adresu – co wiaze sie z ryzykiem wycieku pamieci

Kopiuje łańcuch bez przydzielenia mu obszaru pamięci. Powinien użyć operatora new char [len +1] aby alokowac odopwiednia ilosc pamieci.

**2. Wymień trzy problemy pojawiające się w klasach zawierających składowe inicjalizowane wywołaniem operatora new. Wyjaśnij, jak można uniknąć tych problemów.**

a- należy pamiętać o usuwaniu składowych delete w kolejnosci odwrotnej do inicjalizacji, z tym że należy pamiętać o tym czy delete czy delete [], otóż delete [] danego bieku nie powoduje wywołanie automatycznie destruktora obiektu.

b- Tworzenie nowego drugiego obiektu z przydzilanym miejscowa wersją new, wywołanie new zamazuje obiekt poprzedni już istniejący w pamięci wcześniejszej.

C – Jesli tworzymy obiekty z miejscowa wersja new to musimy sami zadbac o wywołanie ich destruktorów.

**3. Jakie metody klas generuje kompilator, kiedy nie zostaną zdefiniowane jawnie? Opisz zachowanie tych niejawnych metod klasy.**

*Konstruktor domyślny* - tworzy konstruktor bez argumentów i nie wykonujący żadnego kodu.

*konstruktor kopiujący* - kopiuje obiekt do nowo utworzonego obiektu tej samej klasy wyłącznie podczas inicjalizacji. Prototyp: nazwa-klasy(const nazwa-klasy &);

StringBad ditto(motto);

StringBad metoo = motto - albo *operator przypisania*

StringBad also = StringBad(motto);

StringBad \* pStringBad = new StringBad(motto);

Deklaracja: nazwa-klasy::nazwa-klasy (const nazwa-klasy & obiekt) {...}

*operator przypisania* - nazwa-klasy & nazwa-klasy::operator=(const nazwa-klasy &);   
podobnie jak operator kopiujący kopiuje wartości składowych pomiędzy obiektami. Statyczne składowe klasy pozostają jednak niezmienione.

*domyślny destruktor*

*operator pobrania adresu* -zwraca adres obiektu na rzecz ktorego zostal wywołany - tzn wartość wskaźnika ***this***

**4. Znajdź i popraw błędy w poniższej deklaracji klasy:**

class nifty

{

//dane

private:

char personality[];

int talents;

// metody

public:

nifty();

nifty(const char \*s);

friend ostream & operator<<(ostream & os, nifty & n);

};

nifty::nifty()

{

personality[0] = '\0';

talents = 0;

}

nifty::nifty(const char \*s)

{

~~personality = new char(strlen(s) + 1)~~

~~personality = s;~~

strcpy(personality, s);

talents = 0;

}

ostream & ~~nifty:~~operator<<(ostream & os, const nifty \*n)

{

os << n.personality << '\n';

os << n.taltents << '\n';

return os;

}

**5. Dana poniższa deklaracja klasy**:

class Golfer

{

private:

char \* fullname;

int games;

int \* scores;

public:

#1Golfer();

#2Golfer(const char \* name, int g = 0);

// tworzy pusta dynamiczna tablice g elementow (o ile g > 0)

#3Golfer( const Golfer & g);

#4~Golfer();

};

**a) Jakie metody klasy zostana wywołane w poniższych instrukcjach?**

Golfer nancy; #1

Golfer lulu("Lelum Polelum"); #2

Golfer roy("Romek", 12); #2

Golfer \* par = new Golfer; # ~~3~~1

Golfer next = lulu; #3

Golfer gazzard = "Spidi"; #2

\*par = nanzy; - operator przypisania

nancy = "Nancy"; #2 a nastepnie operator przypisania ~~błąd - bez~~ operatora przypisania

**b) Widać wyraźnie, że aby klasa była użyteczna, powinna zostać doposażona w szereg metod. Jakich metod wymaga celem zabezpieczenia się przed uszkodzeniem danych obiektów?**

1. Operator przypisania -operator=(const Golfer & g);

2. to co dla 1, ale dla char - Operator=(const char \*)

3. int length() const { return len;} -