**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

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

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