Questão Pai Filha
Assunto avaliado: Polimorfismo.

Um aluno digitou o código C++ a seguir:

```
#include <iostream>
#include <string>
#include <string.h>
#include <stdio.h>

using namespace std;

class Pai
{
  public :
      Pai(){}
     virtual ~Pai(){}

      virtual std::string *toString()
      { std::string *saida = new string("Sou instancia de Pai");
        return saida;
      }

      void toString(std::string *v)
      {
            const std::string *t = toString();
            v->append( *t );
      }

};

class Filha : public Pai
{
public :
      Filha(){}
       virtual ~Filha(){}

      virtual std::string *toString()
      { std::string *saida = new
                std::string("Sou instancia de Filha");
         return saida;
      }
      void toString(std::string *v)
      {     const string *t = toString();
            v->append( *t );
      }
      void toString(char **v)
      {
            string *t = toString();
            char aux;
```

```
            char *listaAux =
                      (char *) malloc(t->length() * sizeof(char));
            for (unsigned int i = 0; i < t->length(); i++)
                 {
                   aux = t->at(i);
                   listaAux[i] = aux;
                 }
            *v = listaAux;
      }
};

void imprime(Pai *q)
{
      std::string t = *q->toString();
      std::cout << "\n" << t << std::endl;
}

int main()
{
  Pai *p = new Pai();
  Pai *q = new Filha();

  std::cout << "\n";
  for (int i = 0; i < 30; i++) std::cout << "-" ;

  imprime(p);

  std::string t2;
  p->toString(&t2);
  std::cout << "Segunda Impressao:" << t2 << endl;

////////////////////////////////////////////////
  std::cout << "\n";
  for (int i = 0; i < 30; i++) std::cout << "-" ;

  imprime(q);

  std::string t3;
  q->toString(&t3);
  std::cout << "Segunda Impressao:" << t3 << endl;

  delete p;
  delete q;
  return 0;
} // fim do programa
```

Analise esse código digitado pelo aluno e em seguida, responda V ou F para as seguintes afirmações.

(V) A resultado da execução do programa imprimirá na tela:

```
------------------------------
```

```
Sou instancia de Pai
Segunda Impressao:Sou instancia de Pai

------------------------------
Sou instancia de Filha
Segunda Impressao:Sou instancia de Filha
```

(F) A resultado da execução do programa imprimirá na tela:

```
------------------------------
Sou instancia de Pai
Segunda Impressao:Sou instancia de Pai

------------------------------
Sou instancia de Pai
Segunda Impressao:Sou instancia de Filha
```

(F) A resultado da execução do programa imprimirá na tela:

```
------------------------------
Sou instancia de Pai
Segunda Impressao:Sou instancia de Pai

------------------------------
Sou instancia de Filha
Segunda Impressao:Sou instancia de Pai
```

(F) O termo *overloading* ou sobrecarga ou *ad hoc polymorphism* é utilizado para descrever situações onde um nome de função tem várias alternativas de implementação. Esse tipo de polimorfismo não ocorre no código digitado pelo aluno para a definição da classe Pai.

(V) O termo *overloading* ou sobrecarga ou *ad hoc polymorphism* é utilizado para descrever situações onde um nome de função tem várias alternativas de implementação. Os métodos da classe Filha "*virtual std::string *toString()*", "`void toString(std::string *v)`" e "`void toString(char **v)`" são exemplos da ocorrência desse tipo de polimorfismo.

(F) O termo *overriding* ou sobrescrita ou *inclusion polymorphism* ocorre na relação de herança entre duas classes quando essas possuem funções com a mesma assinatura. Esse tipo de polimorfismo não ocorre no código digitado pelo aluno.

(F) O termo *polymorphic variable* ou variável polimórfica ou *assignment polymorphism* é uma variável que é declara de um tipo mas que de fato recebe valores de tipos diferentes. Esse tipo de polimorfismo não ocorre no programa digitado pelo aluno.

(V) Se o aluno inserir antes da instrução "`delete p`" do programa principal (main) as seguintes instruções definidas entre aspas

```
char *c = NULL;
```

```
q->toString(&c);
```

, o compilador acusará erro de compilação.

(V) Se o aluno inserir antes da instrução “`delete p`” do programa principal (main) as instruções a seguir, o compilador não acusará erro algum.

```
Filha *f = new Filha();
char *c = NULL;
f->toString(&c);
printf("\nvalor de c >> %s", c);
```

(V) As seguintes instruções, se inseridas no programa principal, irão ocassionar erros de compilação:

```
Pai *z = new Filha();
Filha *w = z;
```

////////////////////////////////

```
/*
 * main.cpp
 *
 *  Created on: Dec 7, 2015
 *      Author: robinsonvidanoronha
 */
#include <iostream>
#include <string>
#include <string.h>
#include <stdio.h>

using namespace std;

class Pai
{
  public :
      Pai(){}
    virtual ~Pai(){}
      virtual std::string *toString()
      { string *saida = new string("Sou instancia de Pai");
     return saida;
```

```
        }

        void toString(std::string *v)
        {
                const string *t = toString();
                v->append( *t );
        }

};

class Filha : public Pai
{
public :
        Filha(){}
     virtual ~Filha(){}
        virtual std::string *toString()
        { string *saida = new string("Sou instancia de Filha");
       return saida;
        }
        void toString(std::string *v)
        {
                const string *t = toString();
                v->append( *t );
        }
        void toString(char **v)
        {
                string *t =  toString();
                char aux;
                char *listaAux = (char *) malloc(t->length() *
sizeof(char));
                for (unsigned int i = 0; i < t->length(); i++)
                        {
                          aux = t->at(i);
                          listaAux[i] = aux;
                        }
                *v = listaAux;
```

```
        }
};

void imprime(Pai *q)
{
        std::string t = *q->toString();
        std::cout << "\n" << t << std::endl;
}
int main()
{
   Pai *p = new Pai();
   Pai *q = new Filha();

   std::cout << "\n";
   for (int i = 0; i < 30; i++) std::cout << "-" ;

   imprime(p);

   std::string t2;
   p->toString(&t2);
   std::cout << "Segunda Impressao:" << t2 << endl;

//////////////////////////////////////////////////
   std::cout << "\n";
   for (int i = 0; i < 30; i++) std::cout << "-" ;

   imprime(q);

   std::string t3;
   q->toString(&t3);
   std::cout << "Segunda Impressao:" << t3 << endl;

   delete p;
   delete q;
   free (c);
   return 0;
}

//////////////////////////////  fim
```