1 A : Determine the exact output of the following program (7 marks):

```
#include <iostream>
using namespace std;

class Food{
      float price;
  protected:
      Food() { price = 0; cout << "F"; }
      void setPrice(const float _p) { price = _p; };
      float getPrice() const { return price; };
      ~Food() { cout << "~F"; }
  public:
      virtual void display(ostream& os) const {
            os << "Food costs: "<< price << endl;
      }
};

class Burger : public Food{
      int patties;
  protected:
      float get() const;
      int get(bool) const;
  public:
      Burger(){ patties = 0; cout << "B"; };
      Burger(int, float);
      Burger(float _pri, int _pat){
            patties = _pat; setPrice(_pri); cout << "B2";
      };
      void setPatties(int);
      virtual void display(ostream&) const;
      ~Burger() { cout << "~B"; }
};

Burger::Burger(int _pat, float _pri){
      patties = 0;
      setPrice(_pri);
      if (_pat < 4)
            patties = _pat;
      cout << "B1";
}

float Burger::get() const{
      return getPrice();
}
int Burger::get(bool _out) const {
      if (_out)
            return patties;
      else
            return 0;
}
void Burger::setPatties(int _pat){
      Burger tempBurger(_pat, getPrice());
      *this = tempBurger;
}
void Burger::display(ostream& os) const{
      os << "Burger patties: " << patties << ", Cost: $" << getPrice() << endl;
}
```

```
ostream& operator<<(ostream& os, const Food& _f){
      _f.display(os);
      return os;
}

int main(){
      int patties=2;
      float price=5.32;
      Burger hb(price,patties);
      cout << "\n--------" << endl;
      cout<<hb;
}
```

1 B : Consider adding the following function calls to the main function in 1A. Which of these calls, if any, would cause a compile or runtime warnings or error (3 marks)?

```
   1) Burger hb2(patties, price);
   2) hb.getPrice();
   3) hb.Food::display(cout);
```

1 C : What does the keyword `protected` in the Food and Burger classes mean (2 marks)?

1 D : List examples of coercion, overloading and inclusion polymorphism in the code in 1A (3 marks).

2 A : Given the following definition of an Apple class, define a Tree class that contains a user-specified number of Apples not greater than 40. Include in your definition:

- **a safe default constructor and**
- **an overloaded constructor that takes as parameters an array of Apples and the number of Apples**

(5 marks)

```
class Apple{
	char colour[10];
	int acidity;
public:
	Apple();
	Apple(const int, const char*);
};
```

2 B : Code the implementation of the constructors of your Tree class (4 marks).

3 A : Consider the following incomplete class definitions. Fill in the missing code as described in the comments. Your solution should call the appropriate set and get functions on a Book object (4 marks).

```
class Book{
public:
	//Provide:
	// - a pure virtual function to set the ISBN/eISBN
	// - a pure virtual function to get the ISBN/eISBN
};

// create an EBook class that inherits from Book
{
	int eISBN;
public:
	void set(const int _eISBN){eISBN=_eISBN;};
	int get() const{return eISBN;};
};
// create a PaperBook class that inherits from Book
{
	int isbn;
public:
	void set(const int _isbn){isbn=_isbn};
	int get() const{return isbn;};
};
```

3 B : Describe in a couple of sentences the purpose of a pure virtual member function (2 marks)

3 C : What is an interface (2 marks)?

4 A : Consider the following function. Write a function template to extend this definition to any fundamental type (4 marks).

```
int equation(int a, int b){
      return int x=2*a+b-5;
}
```

~~**4 B : Specialize your template to receive two C-style character strings. Your specialization should return the address of the longest string. If the strings are of equal length return the address of the first string (4 marks).**~~