Q1. You are given a set of account classes for this questions, in this file Accounts.cpp. There is an abstract base class Account and two concrete child classes, CurrentAccount and LoanAccount.

Using these classes, but without editing them, you will create a new class called AccountsLedger, which can store a set of account objects and perform a number of operations on the accounts in a ledger. The specification and requirements for the AccountsLedger class are as follows:

- An AccountsLedger can store up to 1000 account objects
- When an AccountsLedger is first created, it contains no account objects
- Accounts of either type, CurrentAccount or LoanAccount, can be added to an AccountsLedger by calling an add(...) method on a ledger object. add(...) takes a reference to an account object as a parameter. add(...) returns a bool value to indicate if the account object was successfully added, if there was remaining space in the ledger. Note: You may implement two add(...) methods, one for each account type.
- A ledger has a print() method which, when called, prints out the details of all accounts in the ledger.
- A ledger has an applyCharges(...) method, which takes a given percentage rate as a parameter and deducts that percentage of the balance from each account in the ledger.
- A ledger has a destructor that deallocates any memory that has been allocated using new.

The following test code executed from main()

```
    AccountsLedger ledger;
    ledger.add(LoanAccount(1234,-100000,20));
    ledger.add(LoanAccount(1235,-200000,25));
    ledger.add(LoanAccount(1236,-300000,15));
    ledger.add(CurrentAccount(1237,3000));
    ledger.add(CurrentAccount(1238,1000));
    ledger.add(CurrentAccount(1239,2000));
    ledger.applyCharges(0.01);
    ledger.print();
```

should produce the following printout:

```
account number: 1234 has balance: -101000 Euro - Loan term: 20 months
account number: 1235 has balance: -202000 Euro - Loan term: 25 months
account number: 1236 has balance: -303000 Euro - Loan term: 15 months
account number: 1237 has balance:    2970 Euro - Overdraft limit: 0
account number: 1238 has balance:     990 Euro - Overdraft limit: 0
account number: 1239 has balance:    1980 Euro - Overdraft limit: 0
```

This question is worth 30 of the 100 marks for this assignment. Submit only your AccountsLedger class code.

Q2. A partial implementation of a Complex number class is given below. A Complex object represents a complex number. Complex objects may be added or subtracted using the overloaded operators + and -.

```
class Complex {
private:
    double re;
    double im;
public:
    Complex() : re(0), im(0) {};
    Complex(double r, double i) : re(r), im(i) {};
    Complex operator+(const Complex & c) {
        return Complex(re+c.re, im+c.im);
    }
    Complex operator-(const Complex & c) {
        return Complex(re-c.re, im-c.im);
    }
};
```

Add the following features to the Complex class:

- An overloaded * operator that multiplies two complex numbers
- An overloaded / operator to divide two complex numbers
- An overloaded << operator that allows a Complex object to be passed to cout for printing

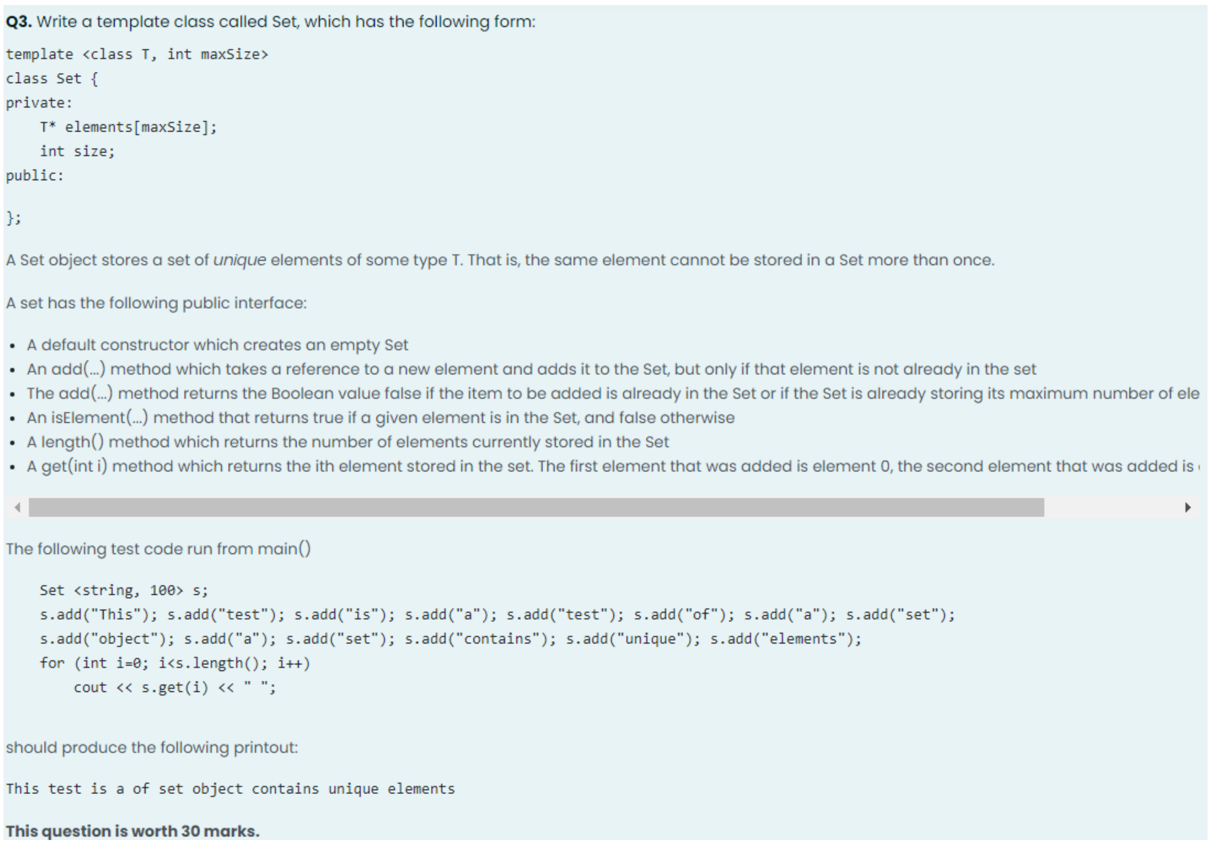
When the following test code is run from main()

```
    Complex c1(1,-2);
    Complex c2(3,-4);
    Complex c3(-3,2);
    cout << (c1 + c2 - c2 - c1);
    cout << endl;
    cout << ((c1-c2)/c3 + c1*c3);
```

your code should produce a printout similar to this:

```
0
1.76923 + 8.15385j
```

This question is worth 20 marks. Submit the completed code for the full Complex class.


Q3. Write a template class called Set, which has the following form:

```
template <class T, int maxSize>
class Set {
private:
    T* elements[maxSize];
    int size;
public:

};
```

A Set object stores a set of *unique* elements of some type T. That is, the same element cannot be stored in a Set more than once.

A set has the following public interface:

- A default constructor which creates an empty Set
- An add(...) method which takes a reference to a new element and adds it to the Set, but only if that element is not already in the set
- The add(...) method returns the Boolean value false if the item to be added is already in the Set or if the Set is already storing its maximum number of ele
- An isElement(...) method that returns true if a given element is in the Set, and false otherwise
- A length() method which returns the number of elements currently stored in the Set
- A get(int i) method which returns the ith element stored in the set. The first element that was added is element 0, the second element that was added is

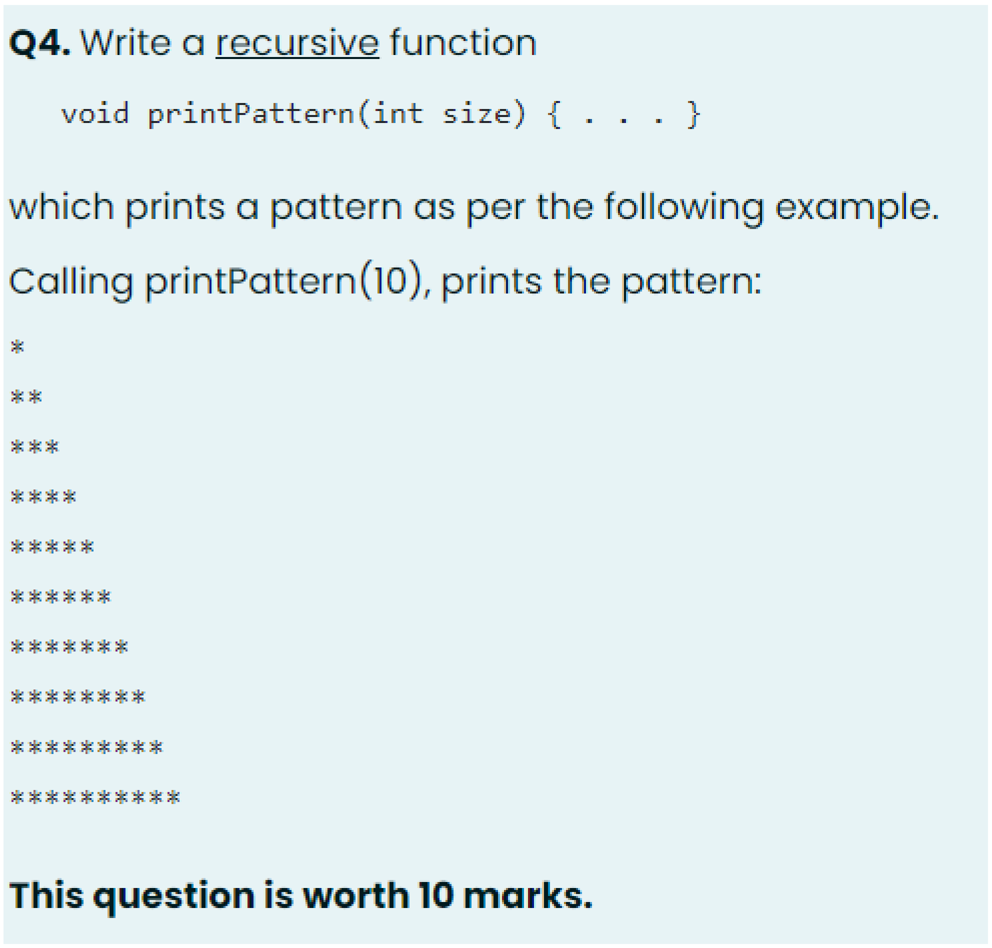
The following test code run from main()

```
    Set <string, 100> s;
    s.add("This"); s.add("test"); s.add("is"); s.add("a"); s.add("test"); s.add("of"); s.add("a"); s.add("set");
    s.add("object"); s.add("a"); s.add("set"); s.add("contains"); s.add("unique"); s.add("elements");
    for (int i=0; i<s.length(); i++)
        cout << s.get(i) << " ";
```

should produce the following printout:

```
This test is a of set object contains unique elements
```

This question is worth 30 marks.


Q4. Write a <u>recursive</u> function

```
   void printPattern(int size) { . . . }
```

which prints a pattern as per the following example.

Calling printPattern(10), prints the pattern:

```
*
**
***
****
*****
******
*******
********
*********
**********
```

This question is worth 10 marks.

Q5. Write a recursive function which returns the value of a given number raised to a positive integer power. Your function will have the following form:

```
   double pow(double a, unsigned int n) { . . . }
```

This question is worth 10 marks.