Midterm Exam

Programming Workshop 2 (CSCI 1061U)

University of Ontario Institute of Technology

April 3, 2018 (Morning)

Total time: 80 minutes

Family name: _____

Given names: _____

Student number: _____

Question	Marks
1	_____ /10
2	_____ /10
3	_____ /30
Total	_____ /50

Instructions

- You are only allowed to use the http://en.cppreference.com/w/ in this exam.
- Please submit your source files and the Makefile via Blackboard.
- Total pages (including the cover page): 6

Question 1 (10 Marks)

Provide `read_arr` and `inplace_rev_arr` function that reverses the contents of an array *in place*. `read_arr` function reads integer values and stores these in an array till termination value is found. In the code below, the termination value is set to `-1`.

```cpp
// filename = q1.cpp

#include <iostream>
#include <vector>

using namespace std;

void print_arr(const vector<int>& a)
{
  vector<int>::const_iterator i;

  for (i = a.begin(); i != a.end(); ++i)
    cout << *i << ' ';
  cout << endl;
}

vector<int> read_arr(int termination)
{
    // TO DO
}

void inplace_rev_arr(vector<int>& a)
{
    // TO DO
}

int main()
{
  vector<int> a = read_arr(-1);

  print_arr(a);
  inplace_rev_arr(a);
  print_arr(a);

  return 0;
}
```

Consider the following text file

```
7
1
-2
3
4
5
7
89
-1
```

When we use this file with this program, the program produces the following output.

```
$ ./q1 < data.txt
7 1 -2 3 4 5 7 89
89 7 5 4 3 -2 1 7
```

Question 2 (10 Marks)

Compute the following code that prints the checker board pattern. We assume that the top-left block is *white*. To keep things civilized, we will represent a white block as `W` and a black block as `_`.

```cpp
// filename = q2.cpp
#include <iostream>

void print_checker_pattern(int height, int width);

int main()
{
  using std::cout;
  using std::cin;

  int height, width;

  cout << "Enter height: "; cin >> height;
  cout << "Enter width: "; cin >> width;

  print_checker_pattern(height, width);

  return 0;
}

void print_checker_pattern(int height, int width)
{
    // TO DO
}
```

The program produces the following output for height and width 8 and 7, respectively.

```
$ ./q2
Enter height: 8
Enter width: 7
W_W_W_W
_W_W_W_
W_W_W_W
_W_W_W_
W_W_W_W
_W_W_W_
W_W_W_W
_W_W_W_
```

Question 3 (30 Marks)

We are given a file that stores information about videos and serials using the following format (first line shows a video and the second line shows a serial; notice that we don't allow for spaces in titles).

```
title: Jumanji : resolution: 1024 x 960 : duration: 90
```

```
title: WestWorld : resolution: 720 x 480 : duration: 23.5 : num_episodes : 45
title: Jumanji1884 : resolution: 1024 x 960 : duration: 90
title: PacificRim : resolution: 1024 x 960 : duration: 90
title: BlackPanther : resolution: 1024 x 960 : duration: 90
title: Dracula : resolution: 1024 x 960 : duration: 90
title: ParksAndRecreation : resolution: 1024 x 960 : duration: 90 : num_episodes : 23
```

Looking at this file we discern the following structure.

Videos

- name (string)
- xres (int)
- yres (int)
- duration (float)

Serials

- name (string)
- xres (int)
- yres (int)
- duration (float)
- num_episods (int)

You are asked to create the Videos and Serials classes, leveraging inheritance. Your code needs to support the following functionality:

- Writing videos and serials to `ostream`. The output must match what is seen in the file above.
- Reading videos and serials from `istream`.
- Ability to search through videos and serials based upon title match. The match doesn't need to be exact.

Consider who we will use this program.

Case 1: When no search keyword is provided

```
$ ./q3 videofile
title: Jumanji : resolution: 1024 x 960 : duration: 90
title: WestWorld : resolution: 720 x 480 : duration: 23.5 : num_episodes : 45
title: Jumanji1884 : resolution: 1024 x 960 : duration: 90
title: PacificRim : resolution: 1024 x 960 : duration: 90
title: BlackPanther : resolution: 1024 x 960 : duration: 90
title: Dracula : resolution: 1024 x 960 : duration: 90
title: ParksAndRecreation : resolution: 1024 x 960 : duration: 90 : num_episodes : 23
```

Case 2: When a search keyword is provided

```
$ ./q3 videofile ji
title: Jumanji : resolution: 1024 x 960 : duration: 90
title: Jumanji1884 : resolution: 1024 x 960 : duration: 90
```

Starter Code

```cpp
#include <iostream>
#include <string>

using namespace std;


class Video
{
protected:
  string _title;
  int _xres, _yres;
  float _duration;

};


class Serial : public Video
{
protected:
  int _num_episodes;
};

// Read videos/serials from istream into an array
// Write video/serials stored in an array to ostream
// Search the array and find the matching items and write those to ostream

int main(int argc, char** argv)
{
  return 0;
}
```

Makefile

Modify the following Makefile as needed and upload with your cpp (q1.cpp, q2.cpp and q3.cpp) files.

```
all: q1 q2 q3

q1: q1.cpp
    g++ q1.cpp -o q1

q2: q2.cpp
    g++ q2.cpp -o q2

q3: q3.cpp
    g++ q1.cpp -o q3

.PHONY:
    clean

clean:
    rm q1
    rm q2
    rm q3
```

All Done

Do not forget to submit q1.cpp, q2.cpp, q3.cpp, and Makefile.