פקולטה: מדעי הטבע
מחלקה: מדעי המחשב
שם הקורס: תכנות מערכות ב
קוד הקורס: 7023010 כל הקבוצות
מועד __**א**___ סמסטר: ___**ב**___ שנה: ___ה'תש"ף
תאריך הבחינה: ט' בתמוז ה'תש"ף, 01/07/2020
משך הבחינה: 3 שעות
שם המרצים: אראל סגל-הלוי, ערן קאופמן

יש לקרוא היטב את כל השאלות לפני שמתחילים לכתוב.

בעמוד הבא ישנו טופס שבו אתם יכולים לבקש הסבר על כל דבר שאתם לא מבינים בשאלון.

אנא תלשו את הטופס, מלאו אותו בכתב ברור, והעבירו אותו למרצה כשהוא נכנס לכיתה.

שימו לב: המרצה יעבור בכיתה רק פעם אחת, לכן מומלץ לקרוא היטב את כל השאלות לפני שמתחילים לכתוב.

יינתן מענק של 5 נקודות לסטודנטים שיכתבו את הפתרון באופן ברור קריא וקל לבדיקה, בפרט:

- השאלות מסודרות לפי הסדר מ-1 עד 8;
- כל שאלה בעמוד אחד במחברת: שאלה 1 בעמוד 3, שאלה 2 בעמוד 4, וכו'.
- הכתב ברור וקריא, בלי מחיקות קשקושים חיצים וטקסט מיותר.

אסור להשתמש בכל חומר עזר.

יש לענות על כל השאלות במחברת הבחינה.

אין צורך להעתיק את השאלון למחברת - השאלון יתפרסם בגיטהאב לאחר הבחינה.

יש לענות תשובות מלאות אך ממוקדות. לא יינתנו נקודות על תשובות עם טקסט מיותר שאינו קשור לנושא.

בשאלות "מה סוג השגיאה?" הכוונה לאחד מהסוגים שדיברנו עליהם בהרצאות, כגון: שגיאת קומפילציה, קישור, ריצה, לוגית, לולאה אינסופית או דליפת-זיכרון.

בהצלחה!!!

---סטודנטים יקרים! כדי לשמור על בריאותכם, המרצה יענה לשאלותיכם מרחוק. ---
--- אנא תלשו דף זה מטופס הבחינה, רשמו בו את שאלותיכם, והעבירו אותו למרצה כשהוא ייכנס לחדר. תודה ---

שם:	(רשות)
דברים לא ברורים בשאלה 1:	
דברים לא ברורים בשאלה 2:	
דברים לא ברורים בשאלה 3:	
דברים לא ברורים בשאלה 4:	
דברים לא ברורים בשאלה 5:	
דברים לא ברורים בשאלה 6:	
דברים לא ברורים בשאלה 7:	
דברים לא ברורים בשאלה 8:	

שאלה 1 [10 נק'] - מתוך האקראנק ++C

תקציר בעברית:

בשאלה זו עליכם לתקן באג בקוד הנתון. הקוד אמור לחשב את הרווח של בית-מלון מהשכרת חדרים ודירות.

- המחירים נתונים ע"י הנוסחה:
מחיר עבור חדר רגיל (standard room) במלון מחושב ע"פ = [מספר חדרי שינה בחדר כפול 50] ועוד [מספר האמבטיות בחדר כפול 100].
- מחיר עבור דירה (apartment) במלון מחושב ע"פ = [מספר חדרי שינה בדירה כפול 50] ועוד [מספר אמבטיות כפול 100] ועוד 100.

הרווח של בית-המלון הוא סכום הרווח שלו מכל החדרים שהוא משכיר.

קלט התוכנית:

בשורה הראשונה של הקלט נמצא מספר החדרים/דירות שהמלון משכיר.

בשורות הבאות, כל שורה מתארת חדר או דירה, מספר חדרי-השינה, ומספר האמבטיות.

לדוגמה, אם הקלט הוא:

```
2
standard 3 1
apartment 1 1
```

אז הפלט הצפוי הוא:

```
500
```

הסבר: המלון מרויח

- 250 מהחדר הרגיל שיש בו 3 חדרי-שינה + 1 אמבטיה,
ועוד
- 250 מהדירה שיש בה 1 חדר-שינה + 1 אמבטיה.

בתוכנית למטה יש באג הגורם לשגיאה לוגית – הפלט המתקבל נמוך מדי. בדוגמה למעלה מתקבל הפלט 400.

א. [5 נק']. הסבירו ממה בדיוק נובעת השגיאה (ציינו מספרי שורות רלבנטיות).
ב. [5 נק']. הסבירו איך לתקן את השגיאה (ציינו מספרי שורות); אין לשנות את התוכנית הראשית.

בהמשך מובא הסבר מלא באנגלית; אם הבנתם את ההסבר בעברית אתם יכולים לדלג עליו ולעבור ישר לקוד.

Hotel Prices

In this challenge, the task is to debug the existing code to successfully execute all provided test files.

The given code defines two classes `HotelRoom` and `HotelApartment` denoting respectively a standard hotel room and a hotel apartment. An instance of any of these classes has two parameters:

`bedrooms` and `bathrooms` denoting respectively the number of bedrooms and the number of bathrooms in the room. The prices of a standard hotel room and hotel apartment are given as:

- Hotel Room: 50 * bedrooms + 100*bathrooms
- Hotel Apartment: The price of a standard room with the same number bedrooms and bathrooms plus 100.

For example, if a standard room costs 200, then an apartment with the same number of bedrooms and bathrooms costs 300.

In hotel's codebase, there is a piece of code reading the list of rooms booked for today and calculates the total profit for the hotel. However, sometimes calculated profits are lower than they should be.

Debug the existing `HotelRoom` and `HotelApartment` classes' implementations so that the existing code computing the total profit returns a correct profit.

Your function will be tested against several cases by the locked template code.

Input Format

The input is read by the provided locked code template.
In the first line, there is a single integer *n* denoting the number of rooms booked for today.
After that *n* lines follow. Each of these lines begins with a `room_type` which is either `standard` or `apartment`, and is followed by the number of bedrooms and the number of bathrooms in this room.

Constraints

- 1 <= n <= 100.
- There is at least 1 and at most 5 bedrooms in a room.
- There is at least 1 and at most 5 bathrooms in a room.

Output Format

The output is produced by the provided and locked code template. It calculates the total profit by iterating through the vector of all rooms read from the input.

Sample Input 0

```
2
standard 3 1
apartment 1 1
```

Sample Output 0

```
500
```

Explanation 0

In the sample we have one standard room with 3 bedrooms and 1 bathroom, and one apartment with 1 bedrooms and 1 bathroom. The price for the room is 3*50+100=250. The price for the apartment is 50+100+100=250. Thus the hotel profit is as the sum of prices of both rooms.

Code

```
#include <iostream>
#include <vector>

using namespace std;

class HotelRoom {
public:
    HotelRoom(int bedrooms, int bathrooms)
    : bedrooms_(bedrooms), bathrooms_(bathrooms) {}

    int get_price() {
        return 50*bedrooms_ + 100*bathrooms_;
    }
private:
    int bedrooms_;
    int bathrooms_;
};

class HotelApartment : public HotelRoom {
public:
     HotelApartment(int bedrooms, int bathrooms)
     : HotelRoom(bedrooms, bathrooms) {}

    int get_price() {
         return HotelRoom::get_price() + 100;
     }
 };

 int main() {
     int n;
     cin >> n;
     vector<HotelRoom*> rooms;
     for (int I = 0; I < n; ++i) {
         string room_type;
         int bedrooms;
         int bathrooms;
         cin >> room_type >> bedrooms >> bathrooms;
         if (room_type == “standard”) {
             rooms.push_back(new HotelRoom(bedrooms, bathrooms));
         } else {
             rooms.push_back(new HotelApartment(bedrooms, bathrooms));
         }
     }

     int total_profit = 0;
     for (auto room : rooms) {
         total_profit += room->get_price();
     }
     cout << total_profit << endl;

     for (auto room : rooms) {
         delete room;
     }
     rooms.clear();

     return 0;
 }
```

שאלה 2 [10 נק'] - מתוך האקראנק לינוקס

תקציר בעברית: כתבו פקודה בלינוקס, המקבלת שורות טקסט דרך הקלט התקני, ומחליפה כל הופעה של המילה "thy" במילה "your".

- אין להבחין בין אותיות גדולות לקטנות – יש להחליף גם THY וגם tHy וכו'.
- יש להחליף מילים שלמות בלבד – למשל את המחרוזת שבתוך xxxTHYzzz אין להחליף כלל.

Task

For each line in a given input file, transform all the occurrences of the word '*thy*' with '*your*'. The search should be **case insensitive**, i.e. '*thy*', '*Thy*', '*tHy*' etc. Should be transformed to '*your*'.

Input Format

A text file will be piped into your command via STDIN.

Output Format

Transform and display the text as required in the task.

Sample Input

The line:

```
"Feed'st thy light's flame with self-substanthyal fuel,"
```

should be transformed to:

```
"Feed'st your light's flame with self-substanthyal fuel,"
```

The line:

```
"Thy self thy foe, to thy sweet self too cruel:"
```

should be transformed to:

```
"your self your foe, to your sweet self too cruel:"
```

שאלה 3 [10 נק']

נתונה מחלקה Board המייצגת לוח במשחק איקס-עיגול, וכן תוכנית ראשית להדגמה:

```
01   #include <iostream>
02   #include <string>
03   #include <cmath>
04   #include <vector>
05   #include <utility>
06
07   using namespace std;
08
09   class Board {
10       vector<vector<char>> board;
11   public:
12       Board():  board(3, vector<char>(3, '.')) {}
13
14       char operator[](pair<int,int> rowcol) const {
15           return board[rowcol.first][rowcol.second];
16       }
17
18       char& operator[](pair<int,int> rowcol)  {
19           return board[rowcol.first][rowcol.second];
20       }
21   };
22
23   int main() {
24       Board b;
25       cout << b[{1,1}] << endl; // should print "."
26       b[{1,1}] = 'X';
27       cout << b[{1,1}] << endl; // should print "X"
28       try {
29           b[{1,1}] = 'R';  // should throw an exception
30       } catch (string message) {
31           cout << message << endl;   // should print “illegal char”
32       }
33       cout << b[{1,1}] << endl; // should print "X"
34   }
```

התוכנית רצה ועובדת, אבל היא לא זורקת שום חריגה; אנחנו רוצים שתיזרק חריגה כשמישהו מנסה להכניס ללוח תו שהוא לא איקס ולא עיגול (למשל בשורה 29). עליכם לשנות את הקוד כך שתיזרק חריגה במקרים אלה, מבלי לשנות את התוכנית הראשית.

רמז: יש להשתמש במחלקה מתווכת ("פרוקסי"). לשנות את שורה 18 כך שהמבנה Board לאחר השינוי ייראה כך:

```
09   class Board {
10       vector<vector<char>> board;
11   public:
12       Board():  board(3, vector<char>(3, '.')) {}
13
```

```
    char operator[](pair<int,int> rowcol) const {
        return board[rowcol.first][rowcol.second];
    }

    char_reference operator[](pair<int,int> rowcol)  {  // ← this line changed
        return board[rowcol.first][rowcol.second];
    }
};
```

מה שנשאר לכם לעשות זה רק לכתוב את המחלקה char_reference.

שאלה 4 [10 נק']

כיתבו תוכנית המקבלת קובץ קלט בשם input.txt המכיל מספרים שלמים חיוביים או שליליים (כל מספר בשורה אחת), וכותבת שני קבצי פלט: positive.txt, negative.txt. הקובץ positive.txt אמור להכיל את כל המספרים החיוביים בקלט, כולל אפס; הקובץ negative.txt אמור להכיל את כל המספרים השליליים בקלט, לא כולל אפס. לדוגמה, אם קובץ הקלט הוא:

Input.txt:

-1
4
55
0
-8
9

אז קבצי הפלט הם:

positive.txt:

4
55
0
9

negative.txt:

-1
-8

כבר כתבנו עבורכם את התוכנית הראשית – אתם צריכים רק להשלים את החסר בשורה האחרונה (המודגשת). אין לבצע שינויים נוספים בקוד.

```
#include <iostream>
#include <fstream>
#include <iterator>
#include <algorithm>
using namespace std;

int main() { // a demo program
    ifstream input("input.txt");
    ofstream positive("positive.txt");
    ofstream negative("negative.txt");
    partition_copy(...);    // COMPLETE THIS LINE
}
```

שימו לב: בנספח לבחינה ישנו דף מהאתר של שפת ++C המסביר על הפונקציה partition_copy. מומלץ להיעזר בו.

שאלה 5 [10 נק']

נתונה התוכנית הבאה:

```
01   #include <iostream>
02   using namespace std;
03
04   class Complex {
05   private:
06       double _re;
07       double _im;
08   public:
09       Complex (double re= 0.0, double im= 0.0): _re(re), _im(im) {    }
10
11       const Complex operator-() const {
12           return Complex(-_re , -_im);
13       }
14
15       const Complex operator+(const Complex& other) const {
16            return Complex(_re + other._re, _im + other._im);
17       }
18
19       Complex& operator+=(const Complex& other) {
20           _re+= other._re;
21           _im+= other._im;
22           return *this;
23       }
24   };
25
26   ostream& operator<< (ostream& os, const Complex& c) {
27       return (os << c._re << '+' << c._im << 'I');
28   }
29
30   int main() {
31       Complex c1(1,2);
32       Complex c2(2,3);
33       cout << c1 + c2 << endl;  // should print "3+5i"
34       return 0;
35   }
```

כשמריצים אותה מתקבלת שגיאת קומפילציה:

```
ComplexDemo.cpp:32:21: error: '_re' is a private member of 'Complex'
    return (os << c._re << '+' << c._im << 'I');
^

ComplexDemo.cpp:11:12: note: declared private here
    double _re;
^
ComplexDemo.cpp:32:37: error: '_im' is a private member of 'Complex'
    return (os << c._re << '+' << c._im << 'I');
^
```

```
ComplexDemo.cpp:12:12: note: declared private here
     double _im;
^
```

א [5 נק']. מהי הדרך הנכונה ביותר (מבחינת הנדסת תוכנה) לתקן את השגיאה? אין לשנות את התוכנית הראשית.

ב [5 נק']. סטודנט ניסה לתקן את השגיאה מהסעיף הקודם ע"י העברת השורות 26-28 לפני שורה 24, כך שהתקבל הקוד הבא (ללא שינוי בתוכנית הראשית):

```
04   class Complex {
05   private:
06       double _re;
07       double _im;
08   public:
09       Complex (double re= 0.0, double im= 0.0): _re(re), _im(im) {    }
10
11       const Complex operator-() const {
12           return Complex(-_re , -_im);
13       }
14
15       const Complex operator+(const Complex& other) const {
16            return Complex(_re + other._re, _im + other._im);
17       }
18
19       Complex& operator+=(const Complex& other) {
20           _re+= other._re;
21           _im+= other._im;
22           return *this;
23       }
24
25       ostream& operator<< (ostream& os, const Complex& c) {
26          return (os << c._re << '+' << c._im << 'I');
27       }
28   };
```

הפעם התקבלה השגיאה הבאה:

```
ComplexDemo.cpp:26:14: error: overloaded 'operator<<' must be a binary
operator (has 3 parameters)
     ostream& operator<< (ostream& os, const Complex& c)}
^
```

מה סוג השגיאה וממה בדיוק היא נובעת? פרטו והסבירו את הודעת השגיאה.

שאלה 6 [10 נק']

נתונה התוכנית הבאה:

```
01   #include <iostream>
02   #include <fstream>
03   using namespace std;
04
05   struct RGB {
06     uint8_t red, green, blue;
07   public:
08     RGB(): red(0), green(0), blue(0) {}
09     RGB(uint8_t r, uint8_t g, uint8_t b): red(r), green(g), blue(b) {}
10   };
11
12   int main() {
13     const int dimx = 800, dimy = 800;
14     ofstream imageFile("image.ppm", ios::out | ios::binary);
15     imageFile << "P6" << endl << dimx <<" " << dimy << endl << 255 << endl;
16     RGB* image[dimx*dimy];
17     for (int j = 0; j < dimy; ++j)  {  // row
18       for (int i = 0; i < dimx; ++i) { // column
19         image[dimx*j+i] = new RGB{255,0,0};
20       }
21     }
22
23     imageFile.write(reinterpret_cast<char*>(&image), 3*dimx*dimy);
24     imageFile.close();
25
26     for (int j = 0; j < dimy; ++j)  {  // row
27       for (int i = 0; i < dimx; ++i) { // column
28         delete image[dimx*j+i];
29       }
30     }
31
32     return 0;
33   }
```

התוכנית אמורה לצייר ציור מסויים ולשים אותו בקובץ `image.ppm`, אבל בפועל כשמריצים את התוכנית ופותחים את הקובץ – מגלים שם ציור אחר לגמרי:

א [5 נק']. מה הציור שאמור להיות בקובץ? הסבירו בפירוט איך בדיוק הציור אמור להגיע לקובץ (ציינו מספרי שורות רלבנטיות).

ב [5 נק']. מדוע הציור בפועל הוא שונה? מה סוג השגיאה וממה בדיוק היא נובעת? פרטו (ציינו מספרי שורות).

שאלה 7 [20 נק']

בשאלה זו נכתוב פונקציה בשם find, אשר מוצאת מילה נתונה בטקסט גם אם סדר האותיות במילה התחלף.

למשל: נניח שהטקסט הוא "tDno rryow eb phapy", אם המילה שצריך לחפש היא "worry", אז הפלט יהיה "rryow" - זו אותה מילה רק בשינוי סדר האותיות (באנגלית זה נקרא אנגרמה – anagram).

ניתן להניח שבמחרוזת יש רק אותיות אנגליות גדולות וקטנות, ורווחים (ללא סימני פיסוק). יש לחפש רק מילים שלמות. יש להבחין בין אותיות גדולות לקטנות (למשל במחרוזת למעלה, נמצא "Dont" אבל לא נמצא "dont".

מקרים מיוחדים:

- אם המילה שמחפשים אינה מופיעה בטקסט - יש לזרוק חריגה.
- אם המילה שמחפשים מופיעה פעמיים או יותר - יש להחזיר את ההופעה *הראשונה*.
- אם המילה שמחפשים היא ריקה, או שהיא לא מילה אחת (מכילה רווח) - יש לזרוק חריגה.

תוכנית ראשית לדוגמה:

```
#include "AnagramFinder.hpp"
#include <iostream>
#include <stdexcept>
using namespace std;

int main() {
      string text = " tDno rryow eb phapy";
      cout << anagram::find(text, "Dont") << endl;    // "tDno"
      cout << anagram::find(text, "worry") << endl;   // "rryow"
      cout << anagram::find(text, "Be") << endl;      // "eb"
      cout << anagram::find(text, "happy") << endl;   // "phapy"
      try {
         cout << anagram::find(text, "dont") << endl;
            // should throw an exception – dont != Dont
      } catch (const exception& ex) {
         cout << "   caught exception: " << ex.what() << endl;
            // should print: "Word 'dont' is not found
      }
      try {
          cout << anagram::find(text, "rry") << endl;
              // should throw an exception – not a whole word
      } catch (const exception& ex) {
          cout << "   caught exception: " << ex.what() << endl;
            // should print: "Word 'rry' is not found
      }
}
```

חלק א [10 נק']. כתבו קובץ המבצע **בדיקות-יחידה** לפונקציה בעזרת מערכת **doctest** שהשתמשתם בה במטלות. יש לכתוב רק את הקובץ הכולל את הבדיקות עצמן (כמו הקובץ שהגשתם בחלק א של המטלה) – אין צורך לכתוב את התוכנית הראשית המריצה את הבדיקות (TestRunner.cpp). אין לחזור על בדיקות שנמצאות בתוכנית ההדגמה – תהיו מקוריים. הוסיפו הסברים והערות בקובץ כדי להסביר מה אתם בודקים.

בשאלה זו מספיק לכתוב 15-20 בדיקות (CHECK) בסה"כ; כמובן יש לכסות את כל הדרישות של המטלה.

חלק ב [10 נק']. נתון מימוש לפונקציה find שבו הושמטו כמה חלקים ובמקומם נכתבו קוים תחתיים. השלימו את החסר בכל קו. אין להעתיק את כל המימוש –יש לכתוב במחברת הבחינה רק את החלקים החסרים, לפי המספר המופיע ליד כל קו. שימו לב: כל קו תחתי יכול לייצג מילה אחת או יותר.

המימוש פועל ע"פ האלגוריתם הבא:

- במילה שמחפשים, סדר את האותיות בסדר עולה (למשל אם מחפשים worry נקבל orrwy);
- עבור על כל המילים בטקסט, מילה אחר מילה. עבור כל מילה בטקסט:
  - סדר את האותיות במילה (למשל המילה rryow תהפוך ל orrwy);
  - אם לאחר הסידור מתקבלת מילה זהה למילה שמחפשים – החזר את המילה מהטקסט לפני הסידור.

```
#include "AnagramFinder.hpp"
#include <sstream>
#include <algorithm>
#include <stdexcept>
using __________1;

namespace __________2 {
    __________3 find(__________4 text, __________5 wanted_word) {
        __________6;
        string text_word, sorted_text_word;
        istringstream text_stream(text);
        while (text_stream >> text_word)  {  // read the words in "text" one by one
            sorted_text_word = text_word;
            sort(__________7);
            if (__________8)
                return __________9;
        }
        __________10;
    }
}
```

שאלה 8 [20 נק']

כיתבו דמוי-מיכל בשם **range** המייצג טווח של מספרים שלמים. הטווח יכול לתמוך **בסוגים שונים של מספרים שלמים**, כגון: short, int, long (אין צורך לתמוך במספרים ממשיים). המספרים יכולים להיות מסודרים בסדר **עולה או יורד**.

לפניכם תוכנית ראשית לדוגמה:

```
#include <iostream>
#include <vector>
#include <string>
#include <ostream>
#include "range.hpp"

using namespace std;
using namespace itertools;

int main() {
      for (auto i: range<short>(5,9))
            cout << i << " ";      // 5 6 7 8
      for (auto i: range<short>(9,5))
            cout << i << " ";      // 9 8 7 6
      for (auto i: range<short>(9,9))
            cout << i << " ";      // Nothing is printed - empty range
      for (auto i: range<long>(111111111111,111111111114))
            cout << i << " ";      // 111111111111 111111111112 111111111113
}
```

עליכם לכתוב רק את הקובץ range.hpp.

נספח –דפים מתוך האתר cplusplus.com

std::partition_copy

```
template <class InputIterator, class OutputIterator1,
          class OutputIterator2, class UnaryPredicate pred>
pair<OutputIterator1,OutputIterator2>
    partition_copy (InputIterator first, InputIterator last,
           OutputIterator1 result_true, OutputIterator2 result_false,
           UnaryPredicate pred);
```

Partition range into two.

Copies the elements in the range `[first,last)` for which *pred* returns `true` into the range pointed by *result_true*, and those for which it does not into the range pointed by *result_false*.

The behavior of this function template is equivalent to:

```
template <class InputIterator, class OutputIterator1,
          class OutputIterator2, class UnaryPredicate pred>
  pair<OutputIterator1,OutputIterator2>
    partition_copy (InputIterator first, InputIterator last,
                    OutputIterator1 result_true, OutputIterator2
result_false,
                    UnaryPredicate pred)
{
  while (first!=last) {
    if (pred(*first)) {
      *result_true = *first;
      ++result_true;
    }
    else {
      *result_false = *first;
      ++result_false;
    }
    ++first;
  }
  return std::make_pair (result_true,result_false);
}
```

Parameters

first, last

Input iterators to the initial and final positions of the range to be copy-partitioned. The range used is `[first,last)`, which contains all the elements between *first* and *last*, including the element pointed by *first* but not the element pointed by *last*.

result_true

Output iterator to the initial position of the range where the elements for which *pred* returns `true` are stored.

result_false

Output iterator to the initial position of the range where the elements for which *pred* returns `false` are stored.

pred

Unary function that accepts an element pointed by `InputIterator` as argument, and returns a value convertible to `bool`. The value returned indicates on which result range the element is copied.

The function shall not modify its argument.
This can either be a function pointer or a function object.

The ranges shall not overlap.
The type pointed by the *output iterator* types shall support being assigned the value of an element in the range `[first,last)`.

Return value

A `pair` of iterators with the end of the generated sequences pointed by *result_true* and *result_false*, respectivelly.

Its member `first` points to the element that follows the last element copied to the sequence of elements for which *pred* returned `true`.
Its member `second` points to the element that follows the last element copied to the sequence of elements for which *pred* returned `false`.

Example

```
// partition_copy example
#include <iostream>     // std::cout
#include <algorithm>    // std::partition_copy, std::count_if
#include <vector>       // std::vector

bool IsOdd (int i) { return (i%2)==1; }

int main () {
  std::vector<int> foo {1,2,3,4,5,6,7,8,9};
  std::vector<int> odd, even;

  // resize vectors to proper size:
  unsigned n = std::count_if (foo.begin(), foo.end(), IsOdd);
  odd.resize(n); even.resize(foo.size()-n);

  // partition:
  std::partition_copy (foo.begin(), foo.end(), odd.begin(), even.begin(), IsOdd);

  // print contents:
  std::cout << "odd: ";  for (int& x:odd)  std::cout << ' ' << x; std::cout << '\n';
  std::cout << "even: "; for (int& x:even) std::cout << ' ' << x; std::cout << '\n';

  return 0;
}
```

Output:

```
odd: 1 3 5 7 9
even: 2 4 6 8
```

std::sort

default (1)	`template <class RandomAccessIterator>` `  void sort (RandomAccessIterator first, RandomAccessIterator last);`
custom (2)	`template <class RandomAccessIterator, class Compare>` `  void sort (RandomAccessIterator first, RandomAccessIterator last, Compare comp);`

Sort elements in range
Sorts the elements in the range `[first,last)` into ascending order.

The elements are compared using `operator<` for the first version, and *comp* for the second.

Equivalent elements are not guaranteed to keep their original relative order (see `stable_sort`).

Parameters

first, last

Random-access iterators to the initial and final positions of the sequence to be sorted. The range used is `[first,last)`, which contains all the elements between *first* and *last*, including the element pointed by *first* but not the element pointed by *last*.
`RandomAccessIterator` shall point to a type for which `swap` is properly defined and which is both ***move-constructible*** and ***move-assignable***.

comp

Binary function that accepts two elements in the range as arguments, and returns a value convertible to `bool`. The value returned indicates whether the element passed as first argument is considered to go before the second in the specific *strict weak ordering* it defines.
The function shall not modify any of its arguments.
This can either be a function pointer or a function object.

Return value

none

Example

```
// sort algorithm example
#include <iostream>     // std::cout
#include <algorithm>    // std::sort
#include <vector>       // std::vector

bool myfunction (int i,int j) { return (i<j); }

struct myclass {
  bool operator() (int i,int j) { return (i<j);}
} myobject;

int main () {
  int myints[] = {32,71,12,45,26,80,53,33};
  std::vector<int> myvector (myints, myints+8);               // 32
71 12 45 26 80 53 33

  // using default comparison (operator <):
```

Edit & Run

```
  std::sort (myvector.begin(), myvector.begin()+4);           //(12 32 45 71)26 80 53 33

  // using function as comp
  std::sort (myvector.begin()+4, myvector.end(), myfunction); // 12 32 45 71(26 33 53 80)

  // using object as comp
  std::sort (myvector.begin(), myvector.end(), myobject);     //(12 26 32 33 45 53 71 80)

  // print out content:
  std::cout << "myvector contains:";
  for (std::vector<int>::iterator it=myvector.begin(); it!=myvector.end(); ++it)
      std::cout << ' ' << *it;
  std::cout << '\n';

  return 0;
}
```

Output:

```
myvector contains: 12 26 32 33 45 53 71 80
```