CSCE 221 Cover Page
Homework #1
Due September 18 at midnight to CSNet

First Name Raymond Last Name Zhu UIN 923008555

User Name rawrbyte E-mail address rawrbyte@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero. According to the University Regulations, Section 42, scholastic dishonesty are including: acquiring answers from any unauthorized source, working with another person when not specifically permitted, observing the work of other students during any exam, providing answers when not specifically authorized to do so, informing any person of the contents of an exam prior to the exam, and failing to credit sources used. Disciplinary actions range from grade penalties to expulsion read more: Aggie Honor System Office

Type of sources		Web page (random seed generator) Stackoverflow	
People			
Web pages (provide URL)		http://www.cplusplus.com/forum/beginner/58529/ http://stackoverflow.com/questions/28376195/removing-elements-from-a-vector-with-o1-runtime	
Printed material			
Other Sources		My_vec.cpp	

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.

"On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work."

Your Name Raymond Zhu Date 08/17/15

Type the solutions to the homework problems listed below using preferably LyX/LaTeX program, see the class webpage for more information about its installation and tutorial.

1. (50 points) There are two players. The first player selects a random number between 1 and 32 and the other one (could a computer) needs to guess this number asking a minimum number of questions. The first player responses possible answers to each question are:

   - *yes* – the number is found
   - *lower* – the number to be guessed is smaller than the number in the question
   - *higher* – the number to be guessed is greater than the number in the question

   Hint. The number of questions in this case (range $[1, 32]$) should not exceed 6.

   (a) Write a C++ code for this algorithm and test it using the following input in ranges from 1 to: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048. Be sure that your program throws an exception in case of an invalid dialog entry during the computations.

   Note that the "brute force" algorithm is not accepted.

```
//main file for 2^n range
#include <iostream>
#include <vector>
#include "My_range.h"
using namespace std;

int main()
{
        try{
        //      vector<int> v;

                vector<int> set = {1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048};
        //      vector<int> set = {1, 3, 7, 15, 31, 63, 127, 255, 511, 1023, 2047};
                int range = 0;
                int num = 0;
                int index = 0;
                int pos = 1;

                char input;

                bool check = false;

                cout << "Input a range of 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, or 2048: ";
        //      cout << "Input a range of 1, 3, 7, 15, 31, 63, 127, 255, 511, 1023, 2047: ";
                cin >> range;

                index = in_index(set, range);

                cout << "Choose a value between the range of 1 - " << set[index] << ": " ;
                cin >> num;

                in_range(set[index], num);

                My_range guess;
                guess.set_range(range);
                guess.set_num(num);
                cout << "Input values: correct (y), low (l), or high (h)" << endl;
                attempt(guess, pos, num, range, input, check);

                cout << "The computer took: " << guess.get_count() << " guesses!" << endl;

                return 0;
                }

        catch (exception& e){
                cerr << "error: " << e.what() << '\n';
                return 1;
        }
        catch (int){
                cerr << "Range Error" << endl;
                return 2;
        }
        catch(...){
                cerr << "Oops: unknown exception!\n";
                return 3;
        }

}
```

```
//header file
#include <ostream>

using namespace std;

class My_range{

        //member variables
        int pos, count, range, num, hold;
        int *ptr;

public:
        //member functions
        My_range();
        ~My_range();
        int get_range() const;
        int get_pos() const;
        int get_count() const;
        int get_num() const;
        void set_range(const int& elem);
        void set_num(const int& elem);
        int get_seed(const int& max, const int& min);
        bool verdict(int p, int r, int ct, char c);

};

        My_range::My_range(){
                range = 0;
                pos = 1;
                count = 0;
                num = 0;
                hold = 0;
                ptr = new int;
        }

        My_range::~My_range(){
                delete ptr;
        }

        void My_range::set_range(const int& elem){
                range = elem;
        }

        void My_range::set_num(const int& elem){
                num = elem;
        }

        int My_range::get_range() const{
                return range;
        }

        int My_range::get_pos() const{
                return pos;
        }

        int My_range::get_count() const{
                return count;
        }

        int My_range::get_num() const{
                return num;
        }

        int My_range::get_seed(const int& max, const int& min){
                num = (max+min)/2;
                ++count;
                return num;
        }

        bool My_range::verdict(int p, int r, int ct, char c){
                if (c == 'l'){
                        if (count == 1){
                                pos = p;
                                range = r;
                                return false;
                        }
                        else if (pos <= p){
                                pos = p+1;
                                return false;
                        }

                }

                else if (c == 'h'){
```

```
                        if (count == 1){

                                pos = 1;
                                range = p;

                                return false;
                        }
                        else if (pos <= p){
                                range = p-1;
                                return false;
                        }

                }
                else if (c == 'y'){
                        return true;
                }
        }

vector<int> vector_range(vector<int> v, int r){
        for(int i=1; i<=r; i++){
                v.push_back(i);
        }

        return v;
}

int in_index(vector<int> set, int r){
        int x = 0;
        int index = 0;
        for(int j=0; j<set.size(); j++){
                if(r != set[j]){
                        ++x;
                }
                if(r == set[j]){
                        index = j;
                }
        }

        if(x>11){ throw x; }
        return index;
}

void in_range(int r, int num){

        if (num <= 0 || num > r){
                throw num;
        }
}

void print_vector(vector<int> v, int r){
        for(int i=0; i<r; i++){
                cout << v[i] << " ";
        }
}

void attempt(My_range& guess, int pos, int num, int range, char input, bool check){
        while (check == false){
        pos = guess.get_seed(guess.get_range(), guess.get_pos());
                cout << "The computer guessed the value: " << pos << endl;
                if(pos == num){
                        cout << "It is equal to your target number: " << num << endl;
                }
        cout << "The number compared to " << num << " is: ";
                cin >> input;
        check = guess.verdict(pos, range, guess.get_count(), input);
        }

}
```

(a) Your program must allow the user to set a target number, so that you can do controlled testing. It would be good if you also had a mode where the human's and computer's roles were switched, with the computer generating a (random) target number and the user trying to guess it.

```
// main file for computer generated guess seed and range
#include <iostream>
#include <vector>
#include <ctime>
#include "My_range.h"
using namespace std;

int main()
```

```
{
        try{
        //      vector<int> v;
                srand(time(NULL));
                vector<int> set = {1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048};

                int range = 0;
                int num = 0;
                int index = 0;
                int pos = 1;

                int input;

                bool check = false;


                My_range guess;
                index = guess.get_seed(12, 1);
                range = set[index];
                in_index(set, index);
                num = guess.get_seed(range, 1);
                in_range(range, pos);

                attempt(guess, pos, num, range, input, check);
                cout << "It took you: " << guess.get_count() << " guesses!" << endl;



                return 0;
                }

        catch (exception& e){
                cerr << "error: " << e.what() << '\n';
                return 1;
        }
        catch (int){
                cerr << "Range Error" << endl;
                return 2;
        }
        catch(...){
                cerr << "Oops: unknown exception!\n";
                return 3;
        }

}

// header file
#include <ostream>

using namespace std;

class My_range{

        //member variables
        int pos, count, range, num, hold;
        int *ptr;

public:
        //member functions
        My_range();
        ~My_range();
        int get_range() const;
        int get_pos() const;
        int get_count() const;
        int get_num() const;
        void set_range(const int& elem);
        void set_num(const int& elem);
        int get_seed(const int& max, const int& min);
        bool verdict(int p, int r, int i);

};

        My_range::My_range(){
                range = 0;
                pos = 1;
                count = 0;
                num = 0;
                hold = 0;
                ptr = new int;
        }

        My_range::~My_range(){
                delete ptr;
        }
```

```
        void My_range::set_range(const int& elem){
                range = elem;
        }

        void My_range::set_num(const int& elem){
                num = elem;
        }

        int My_range::get_range() const{
                return range;
        }

        int My_range::get_pos() const{
                return pos;
        }

        int My_range::get_count() const{
                return count;
        }

        int My_range::get_num() const{
                return num;
        }

        int My_range::get_seed(const int& max, const int& min){
                num = rand() % (max - min) + min;
                count = 1;
                return num;
        }

        bool My_range::verdict(int p, int r, int i){
                count++;
                if (i < p){
                                cout << "low" << endl;
                                return false;
                }

                else if (i > p){
                                cout << "high" << endl;
                                return false;
                }

                else if (i == p){
                        cout << "correct" << endl;
                        return true;
                }
        }

vector<int> vector_range(vector<int> v, int r){
        for(int i=1; i<=r; i++){
                v.push_back(i);
        }

        return v;
}

void in_index(vector<int> set, int i){
        int x = 0;
        int index = 0;
        for(int j=0; j<set.size(); j++){
                if(set[i] != set[j]){
                        ++x;
                }
        }
        if(x>12){ throw x; }

}

void in_range(int r, int num){

        if (num <= 0 || num > r){
                throw num;
        }
}

void print_vector(vector<int> v, int r){
        for(int i=0; i<r; i++){
                cout << v[i] << " ";
        }
}

void attempt(My_range& guess, int pos, int num, int range, int input, bool check){
        while (check == false){
```

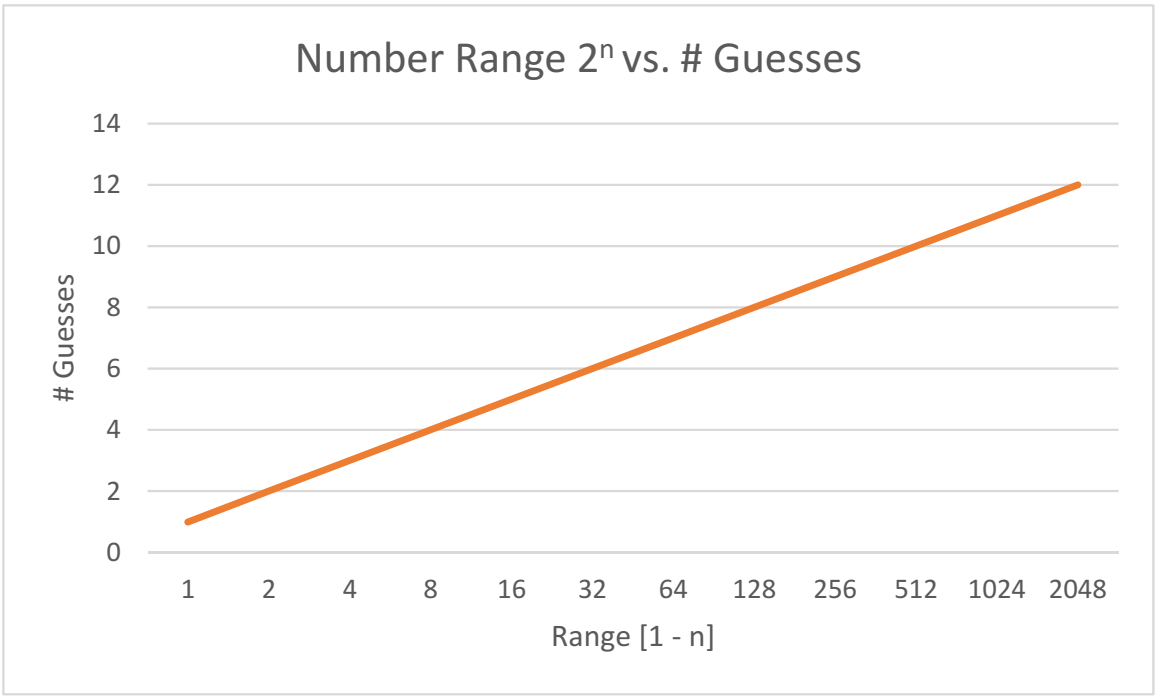
```
                cout << "Pick a value between the range 1 - " << range <<": ";
                        cin >> input;
                in_range(range, input);

                cout << "You guessed the value: " << input << endl;
                if(input == num){
                        cout << "Your guess is equal to the target value" << endl;
                        break;
                }
        cout << "The number is: ";
        check = guess.verdict(num, range, input);
        }

}
```

(a) For the report, you need to measure how many guesses the program takes to find the numbers 2^n and $2^n - 1$ as sample input, not as the only valid input.

i.

Range [1 ... n]	True Answer n	# guesses/comparison	Result of formula in (c)
[1, 1]	1	1	1
[1, 2]	2	2	2
[1, 4]	4	3	3
[1, 8]	8	4	4
[1, 16]	16	5	5
[1, 32]	32	6	6
[1, 64]	64	7	7
[1, 128]	128	8	8
[1, 256]	256	9	9
[1, 512]	512	10	10
[1, 1024]	1024	11	11
[1, 2048]	2048	12	12

```
[rawrbyte]@sun ~/CSCE221/hw1/q1> (15:00:55 09/18/15)
:: ./a.out
Input a range of 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, or 2048: 2048
Choose a value between the range of 1 - 2048: 2048
Input values: correct (y), low (l), or high (h)
The computer guessed the value: 1024
The number compared to 2048 is: l
The computer guessed the value: 1536
The number compared to 2048 is: l
The computer guessed the value: 1792
The number compared to 2048 is: l
The computer guessed the value: 1920
The number compared to 2048 is: l
The computer guessed the value: 1984
The number compared to 2048 is: l
The computer guessed the value: 2016
The number compared to 2048 is: l
The computer guessed the value: 2032
The number compared to 2048 is: l
The computer guessed the value: 2040
The number compared to 2048 is: l
The computer guessed the value: 2044
The number compared to 2048 is: l
The computer guessed the value: 2046
The number compared to 2048 is: l
The computer guessed the value: 2047
The number compared to 2048 is: l
The computer guessed the value: 2048
It is equal to your target number: 2048
The number compared to 2048 is: y
The computer took: 12 guesses!
```



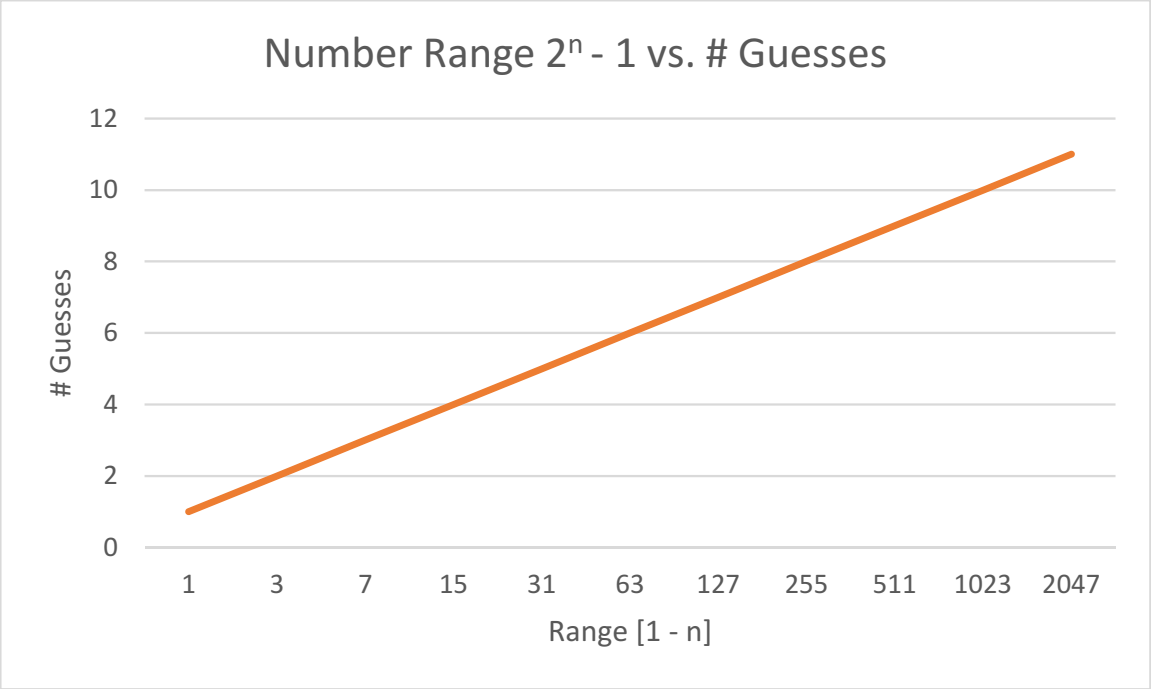
- Formula: Max guesses $= log_2(n) + 1$

(a)

Range [1 ... n]	True Answer n	# guesses/comparison	Result of formula in (c)
[1, 1]	1	1	1
[1, 3]	3	2	2
[1, 7]	7	3	3
[1, 15]	15	4	4
[1, 31]	31	5	5
[1, 63]	63	6	6
[1, 127]	127	7	7
[1, 255]	255	8	8
[1, 511]	511	9	9
[1, 1023]	1023	10	10
[1, 2047]	2047	11	11

```
[rawrbyte]@sun ~/CSCE221/hw1/q1> (14:59:23 09/18/15)
:: ./a.out
Input a range of 1, 3, 7, 15, 31, 63, 127, 255, 511, 1023, 2047: 2047
Choose a value between the range of 1 - 2047: 2047
Input values: correct (y), low (l), or high (h)
The computer guessed the value: 1024
The number compared to 2047 is: l
The computer guessed the value: 1535
The number compared to 2047 is: l
The computer guessed the value: 1792
The number compared to 2047 is: l
The computer guessed the value: 1920
The number compared to 2047 is: l
The computer guessed the value: 1984
The number compared to 2047 is: l
The computer guessed the value: 2016
The number compared to 2047 is: l
The computer guessed the value: 2032
The number compared to 2047 is: l
The computer guessed the value: 2040
The number compared to 2047 is: l
The computer guessed the value: 2044
The number compared to 2047 is: l
The computer guessed the value: 2046
```

```
The number compared to 2047 is: l
The computer guessed the value: 2047
It is equal to your target number: 2047
The number compared to 2047 is: y
The computer took: 11 guesses!
```



- Formula: Max guesses = $\lfloor log_2(n) \rfloor + 1$

```
// main file for 2^(n) - 1
#include <iostream>
#include <vector>
#include "My_range.h"
using namespace std;

int main()
{
        try{
        //      vector<int> v;

        //      vector<int> set = {1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048};
                vector<int> set = {1, 3, 7, 15, 31, 63, 127, 255, 511, 1023, 2047};
                int range = 0;
                int num = 0;
                int index = 0;
                int pos = 1;

                char input;

                bool check = false;

        //      cout << "Input a range of 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, or 2048: ";
                cout << "Input a range of 1, 3, 7, 15, 31, 63, 127, 255, 511, 1023, 2047: ";
                cin >> range;

                index = in_index(set, range);

                cout << "Choose a value between the range of 1 - " << set[index] << ": " ;
                cin >> num;

                in_range(set[index], num);

                My_range guess;
                guess.set_range(range);
                guess.set_num(num);
                cout << "Input values: correct (y), low (l), or high (h)" << endl;
                attempt(guess, pos, num, range, input, check);

                cout << "The computer took: " << guess.get_count() << " guesses!" << endl;
```

```
                return 0;
                }

        catch (exception& e){
                cerr << "error: " << e.what() << '\n';
                return 1;
        }
        catch (int){
                cerr << "Range Error" << endl;
                return 2;
        }
        catch(...){
                cerr << "Oops: unknown exception!\n";
                return 3;
        }

}

// header file
#include <ostream>

using namespace std;

class My_range{

        //member variables
        int pos, count, range, num, hold;
        int *ptr;

public:
        //member functions
        My_range();
        ~My_range();
        int get_range() const;
        int get_pos() const;
        int get_count() const;
        int get_num() const;
        void set_range(const int& elem);
        void set_num(const int& elem);
        int get_seed(const int& max, const int& min);
        bool verdict(int p, int r, int ct, char c);

};

        My_range::My_range(){
                range = 0;
                pos = 1;
                count = 0;
                num = 0;
                hold = 0;
                ptr = new int;
        }

        My_range::~My_range(){
                delete ptr;
        }

        void My_range::set_range(const int& elem){
                range = elem;
        }

        void My_range::set_num(const int& elem){
                num = elem;
        }

        int My_range::get_range() const{
                return range;
        }

        int My_range::get_pos() const{
                return pos;
        }

        int My_range::get_count() const{
                return count;
        }

        int My_range::get_num() const{
                return num;
        }

        int My_range::get_seed(const int& max, const int& min){
                num = (max+min)/2;
```

```
                ++count;
                return num;
        }

        bool My_range::verdict(int p, int r, int ct, char c){
                if (c == 'l'){
                        if (count == 1){
                                pos = p;
                                range = r;
                                return false;
                        }
                        else if (pos <= p){
                                pos = p+2;
                                return false;
                        }

                }

                else if (c == 'h'){
                        if (count == 1){

                                pos = 1;
                                range = p;

                                return false;
                        }
                        else if (pos <= p){
                                range = p-2;
                                return false;
                        }

                }
                else if (c == 'y'){
                        return true;
                }
        }

vector<int> vector_range(vector<int> v, int r){
        for(int i=1; i<=r; i++){
                v.push_back(i);
        }

        return v;
}

int in_index(vector<int> set, int r){
        int x = 0;
        int index = 0;
        for(int j=0; j<set.size(); j++){
                if(r != set[j]){
                        ++x;
                }
                if(r == set[j]){
                        index = j;
                }
        }

        if(x>10){ throw x; }
        return index;
}

void in_range(int r, int num){

        if (num <= 0 || num > r){
                throw num;
        }
}

void print_vector(vector<int> v, int r){
        for(int i=0; i<r; i++){
                cout << v[i] << " ";
        }
}

void attempt(My_range& guess, int pos, int num, int range, char input, bool check){
        while (check == false){
        pos = guess.get_seed(guess.get_range(), guess.get_pos());
                cout << "The computer guessed the value: " << pos << endl;
                if(pos == num){
                        cout << "It is equal to your target number: " << num << endl;
                }
        cout << "The number compared to " << num << " is: ";
                cin >> input;
        check = guess.verdict(pos, range, guess.get_count(), input);
```

```
        }
}
```

(a) Use Big-O asymptotic notation to classify this algorithm.

- $O(log_2 n)$

Submit for grading an electronic copy of your code and solutions to the questions above.

Points Distribution

(a) (b) (5 pts) # guesses in a table; (5 pts) A plot in the report; (15 pts) Program code using STL vector and exception

(c) (5 pts) A math formula of n; (5 pts) Formula results compared to # guesses

(d) (5 pts) A math formula of N; (5 pts) Program code (and the second table)

(e) (5 pts) A big-O function

Submit an electronic copy of your code and results of all your experiments for grading.

2. (15 points) Write a C++ function using the STL string which can determine if a string s is a palindrome, that is, it is equal to its reverse. For example, "racecar" and "gohangasalamiimalasagnahog" are palindromes. Provide 7 test cases, including: the empty string, 4 string which are palindromes and two string which are not palindromes. Write the running time function in terms of n, the length of the string, and its big-O notation to represent the efficiency of your algorithm. Submit an electronic copy of your code and results of all your experiments for grading.

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

string a;
string b = "aibohphobia";
string c = "civic";
string d = "adinida";
string e = "tattarrattat";
string f = "computer";
string g = "science";

void test(string input, string z){
        for (int i = input.length()-1; i >= 0; i--){
                z = z + input[i];
        }


        if(z == input) { cout << z << " is a palindrome!" << endl; }
        else if (z != input) { cout << input << " is NOT a palindrome!" << endl; }

}

int main()
{
        try{

        bool check = false;

        test(b, a);
        test(c, a);
        test(d, a);
        test(e, a);
        test(f, a);
        test(g, a);


                return 0;
        }
        catch (exception& e){
                cerr << "error: " << e.what() << '\n';
                return 1;
        }
        catch(...){
                cerr << "Oops: unknown exception!\n";
                return 2;
        }

}


[rawrbyte]@sun ~/CSCE221/hw1/q2> (14:51:47 09/18/15)
:: ./a.out
aibohphobia is a palindrome!
civic is a palindrome!
adinida is a palindrome!
tattarrattat is a palindrome!
computer is NOT a palindrome!
science is NOT a palindrome!
```

$f(n) = 3n + 2n = 5n = O(f(n) = O(n)$

3. (10 points) Write a function (in pseudo code) which takes as an input an object of `vector` type and removes an element at the rank `k` in the constant time, O(1). Assume that the order of elements does not matter.

   - Assuming that the order of elements does not matter.
   - Consider a vector of integers with size 7
   - vec = {0, 1, 2, 3, 4, 5, 6};

- and you want to remove the 3. You can turn this into
- <0, 1, 2, 6, 4, 5>
- by swapping the last vector position with the 3 and then removing the last vector index
- swap(vec.back(), vec[rank]);
- vec.pop_back();
- in $O(1)$ without any issues.

4. (10 points) **(R-4.39 p. 188)** Al and Bob are arguing about their algorithms. Al claims his $O(n\log n)$-time method is always faster than Bob's O(n^2)- time method. To settle the issue, they perform a set of experiments. To Al's dismay, they find that if $n < 100$, the O(n^2)-time algorithm runs faster, and only when $n \geq 100$ is the O($n\log n$)-time one better. Explain how this is possible.

   - One possible situtation is that: Al: $f(n) = nlog(n) + 962 = nlog(n) + 9216 = O(nlog(n))$ Bob: $f(n) = n_2 = O(n^2)$.
   - When n<100, say n = 99, Al = $99 * log(99) + 96\hat{}2 = 9872.306$, Bob = 99 * 99 = 9801.
   - So Bob is better than Al.
   - When n > 100, say n = 100, Al = 100 *$log(100) + 96^2 = 9880.385$, Bob = 100 x 100 = 100,00.
   - Therefore Al is better than Bob.

5. (15 points) Find the running time functions and classify the algorithms using Big-O asymptotic notation presented in the exercise 4.4, p. 187.

```
Algorithm Ex1(A):
  Input: An array A storing n ≥ 1 integers.
  Output: The sum of the elements at even cells in A.
s ← A[0]
for i ← 2 to n − 1 by increments of 2 do
    s ← s + A[i]
return s
```

$f(n) = \frac{(n-2)}{2} * 2 + 1 = n\ -1 = O(n)$

```
Algorithm Ex2(A):
    Input: An array A storing n ≥ 1 integers.
    Output: The sum of the prefix sums in A.
s ← 0
for i ←0 to n − 1 do
    s ← s + A[0]
    for j ← 1 to n do
        s ← s + A[j]
return s
```

$f(n) = n * nk + 1 = (n^2)k + 1 = O(n^2)$

```
Algorithm Ex2(A):
    Input: An array A storing n ≥ 1 integers.
    Output: The sum of the prefix sums in A.
s ← 0
for i ←0 to n − 1 do
    s ← s + A[0]
    for j ← 1 to i do
        s ← s + A[j]
return s
```

$f(n) = 4 * \frac{n(n-1)}{2} + 1 = 2(n^2 - n) = 2n^2 - 2n = O(n^2)$