Homework 7

● Graded

Student

YADONG WU

View or edit group

Total Points

50 / 50 pts

Question 1

(no title) **6** / 6 pts

1.1 **(no title)** **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

1.2 **(no title)** **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

1.3 **(no title)** **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

1.4 **(no title)** **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

1.5 **(no title)** **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

1.6 **(no title)** **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

Question 2

(no title) **5** / 5 pts

2.1 (no title) **2** / 2 pts

✓ **+ 2 pts** Correct

+ 0 pts Incorrect

2.2 (no title) **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

2.3 (no title) **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

2.4 (no title) **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

Question 3

Big O notation **3** / 3 pts

3.1 (no title) **1** / 1 pt

+ 0 pts Incorrect

✓ **+ 1 pt** Correct

3.2 (no title) **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

3.3 (no title) **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

Question 4

Estimate time complexity **8** / 8 pts

4.1 (no title) **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

4.2 (no title) **2** / 2 pts

+ 0 pts Incorrect

✓ **+ 2 pts** Correct

4.3 (no title) **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

4.4 (no title) **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

4.5 (no title) **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

4.6 (no title) **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

4.7 (no title) **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

Question 5

(no title) **14** / 14 pts

+ 0 pts No submission

✓ **+ 14 pts** correct

+ 10 pts Click here to replace this description.

+ 7 pts Click here to replace this description.

Question 6

(no title) **14** / 14 pts

✓ **+ 14 pts** Correct

+ 0 pts Incorrect

+ 4 pts Click here to replace this description.

+ 8 pts Click here to replace this description.

Q1
6 Points

Q1.1
1 Point

What is the basic idea behind recursion in C++?

- ○ To repeat the same action multiple times
- [✓ To divide a problem into smaller sub-problems and solve each sub-problem individually]
- ○ To write code that repeats a loop until a certain condition is met
- ○ To avoid using loops

Q1.2
1 Point

What is the typical structure of a recursive function?

- ○ A condition that checks if the base case has been reached and the function terminates
- ○ A call to itself that solves a sub-problem
- ○ A loop that repeats the same code multiple times
- [✓ "A condition that checks if the base case has been reached and the function terminates" and "a call to itself that solves a sub-problem"]

Q1.3
1 Point

What is mutual recursion in C++?

- ○ A recursive function that calls itself
- [✓ Two or more recursive functions that call each other]
- ○ A recursive function that calls other non-recursive functions
- ○ A function that calls itself and other non-recursive functions

Q1.4
1 Point

Which of the following is NOT a characteristic of tail recursion?

- ○ The function calls itself as the last operation before returning.
- ○ The function's return value is the same as the return value of the recursive call.
- [✓ The function accumulates its result in a loop.]
- ○ The function performs the same operation in each recursive step.

Q1.5
1 Point

Which of the following return statement implemented the tail recursion? (select all that apply)

- ☐ `return n * factorial(n - 1);` in function `int factorial(int n)`
- [✓ `return factorial(n - 1, result);` in function `int factorial(int n, int result)`]
- ☐ `return fibonacci(n- 1) + fibonacci(n - 2);` in function `int fibonacci(int n )`
- ☐ `return isOdd(n - 1);` in function `bool isEven(int n)`

Q1.6
1 Point

what is the output of the following code?

```
#include <iostream>
void odd(int n);
void even(int n);
void odd(int n) {
  if (n == 0) {
    return;
  } else {
    std::cout << n << " ";
    even(n - 1);
  }
}
void even(int n) {
  if (n == 0) {
    return;
  } else {
    std::cout << n << " ";
    odd(n - 1);
  }
}
int main() {
  odd(3);
  return 0;
}
```

● 3 2 1

○ 3

○ 1

Q2
5 Points

Here is a recursive function for finding all vowels (a, e, i, o, u) in a string:

```
string all_vowels(string str)
{
  if (str.length() == 0) { return ""; } // No vowels
  char first = str[0];
  string rest = str.substr(1);
  string rest_vowels = all_vowels(rest);
  if (is_vowel(first)) { return first + rest_vowels; }
  else { return rest_vowels; }
}
```

Q2.1
2 Points

Based on the above code, what is the function signature of `is_vowel`?

- ○ `void is_vowel(char);`
- [✓ `bool is_vowel(char);`]
- ○ `bool is_vowel(void);`
- ○ `char is_vowel(char);`

Try to write down the function `is_vowel`, which will help you to test the answer to the following questions.

Q2.2
1 Point

What should be the output of the following code

```
int main()
{
 string vowels= all_vowels("baba");
 cout<<vowels<<endl;
 return 0;
}
```

- [✓ aa]
- baba
- bb
- vowels

Q2.3
1 Point

However, it is inefficient to generate a new string in each recursive call. Therefore, we consider writing a recursive helper function

```
string all_vowels(string str, int pos, string already_found)
{
//some statements
}
```

Rearrange the following lines of code to produce a recursive helper function that avoids that problem.

```
1. if (pos == str.length()) { return already_found; }

2. char ch = str[pos];

3. if (is_vowel(ch)) { return all_vowels(str, pos + 1, already_found+ch);}

4. if (is_vowel(ch)) { return all_vowels(str, pos, already_found+ch);}

5. if (is_vowel(ch)) { return all_vowels(str, pos + 1, already_found);}

6. else {return all_vowels(str, pos + 1, already_found+ch);}

7. else {return all_vowels(str, pos + 1, already_found);}
```

Please fill the line numbers in the correct order and separate each number with a comma(for example `1,2,3,4`)

1,2,3,7

Remark:
1 Not all lines are useful.
2 **Extra white space in the answer will be graded as incorrect by the autograder.**
Please make sure there is no extra white space before the first character and after the last character.

Q2.4
1 Point

How should the non-recursive function

```
string all_vowels(string str)
```

call the recursive helper function of the preceding problem?

- ○ return all_vowels(str);
- ○ return all_vowels(str, 0);
- ○ return all_vowels(str, 0, str.length() - 1);
- [✓ return all_vowels(str, 0, "");]

Q3 Big O notation

3 Points

Please identify the leading order term in the following expression when $n \to \infty$:

Q3.1

1 Point

$n^2 + n\log(n)^4 + n + 2 + b\log(n)$, where b is a constant.

- ○ $n\log(n)^4$
- ○ n
- [✓ ● n^2]
- ○ $b\log(n)$

Q3.2

1 Point

$\exp(n) + n^n + n^{10000} + \log(n)^n$, where b is a constant.

- ○ $\exp(n)$
- [✓ ● n^n]
- ○ n^{10000}
- ○ $\log(n)^n$

Q3.3
1 Point

Which of the following calculations is correct, select all that apply

- [✓ $nO(n^2) = O(n^3)$]
- [✓ $\log(n)^{50} + O(n^2) = O(n^2)$]
- [✓ $100 * n^2 + O(n^2) = O(n^2)$]
- $\log(n) * O(n^2) = O(n^2)$
- $n! + O(n^{50}) = O(n^{50})$

Q4 Estimate time complexity
8 Points

Suppose we have a function `int example1(int n)`, which takes one parameter n.

Q4.1
1 Point

If we know the following information:
When n=100, the CPU time cost for running function `example` is 11s.
When n=1000, the CPU time cost for running function `example` is 97s.
When n=10000, the CPU time cost for running function `example` is 995s.
what is the best estimation of the time complexity of function `example` ?

- ○ $O(\log(n))$
- ○ $O(\log(n)n)$
- [✓ $O(n)$]
- ○ $O(n^2)$

Q4.2
2 Points

If we know the following information:
When n=100, the CPU time cost for running function `example` is 11s.
When n=1000, the CPU time cost for running function `example` is 23s.
When n=10000, the CPU time cost for running function `example` is 35s.
what is the best estimation of the time complexity of function `example` ?

- [✓ $O(\log(n))$]
- ○ $O(\log(n)n)$
- ○ $O(n)$
- ○ $O(n^2)$

Q4.3
1 Point

If we know the time complexity of function `int example1(int n)` is $O(n)$ and we know the time cost is 10s when n is 10, the time cost is 100s when n is 100. What is a reasonable estimation of the time cost when n is 400.

- ○ 100
- ○ 200
- [✓ 400]
- ○ 1000

Q4.4
1 Point

If we know the time complexity of function `int example1(int n)` is $O(n^2)$ and we know the time cost is 100s when n is 100, the time cost is 401s when n is 200, the time cost is 890s when n is 300. What is a reasonable estimation of the time cost when n is 1000?

- ○ 1000
- ○ 2000
- ○ 4000
- [✓ 10000]

Q4.5
1 Point

If the time complexity of function `example` is $O(n)$, then what is the time complexity of the following function

```
void fun(int n)
{
        for (int i = 1; i < n;i++)
        {
                example(n);
        }

};
```

- ○ $O(\log(n))$
- ○ $O(\log(n)n)$
- ○ $O(n)$
- [✓ $O(n^2)$]

Q4.6
1 Point

What is the big-O running time of the following algorithm to find an element in an n × n array?

```
for (int i = 0; i < n; i++)
{
  for (j = 0; j < n; j++)
  {
     if (a[i][j] == value) { return true; }
  }
}
```

- ○ $O(n)$
- [✓ $O(n^2)$]
- ○ $O(n \log n)$
- ○ $O(1)$

Q4.7
1 Point

What is the big-O efficiency of reversing all elements in an array of length n, using the following pseudocode?

```
i = 0
j = n - 1
While i < j
{  Swap a[i] and a[j]
   i++
   j--
}
```

- [✓ $O(n)$]
- $O(n^2)$
- $O(n \log n)$
- $O(1)$

Q5
14 Points

Please review the section template in the lecture notes.

Please implement the class `Pair` in `pair.h`, which includes the private member `first` and `second`.

Implement the `sum_diff` member function that returns a pair consisting of the sum of the first and second element of the pair, and the difference of those elements. For example, if the input is 3 and 4, then the output should be 7 and -1. If the input is 4 and 3, then the output should be 7 and 1. Please return a temporary object and do not modify the member variable of the current object.

The elements could be `int`, `double` and `string`. (So you need to use template class). If the first and second elements are string variables. Then define the sum of them to be the concatenation of them, and define the difference of them to be an empty string variable. For example, if the input is "ab" and "cd", then the output is "abcd" and "".
hint: You need to handle this special case with the template specialization.

Please don't add the following statement in the header file

```
using namespace std;
```

Remark: without this statement, you may need to modify your code. For example, you need to change `cout` to `std::cout`, and change `string` to `std::string`.

In the file "HW7P1.cpp", please include the header file "pair.h" and the main function. The first part in `main()` function is given as follows. You need to identify the extra member functions in the class `Pair` and their function signatures based on this snippet.

```
Pair<int> pair1(3, 4);
Pair<int> result1 = pair1.sum_diff();
cout << result1.get_first() << " " << result1.get_second() << endl;
cout << "Expected: 7 -1" << endl;
Pair<double> pair2(3.9, 2.1);
Pair<double> result2 = pair2.sum_diff();
cout << result2.get_first() << " " << result2.get_second() << endl;
Pair<string> pair3("x2", "y2");
Pair<string> result3 = pair3.sum_diff();
cout << result3.get_first() << " " << result3.get_second() << endl;
```

After this part, please add a `cout` statement to display the date, your name, or the name of all group members if you use group submission. Run your code and save the screenshots of the console.

Please upload "pair.h", "HW7P1.cpp" and screenshots here

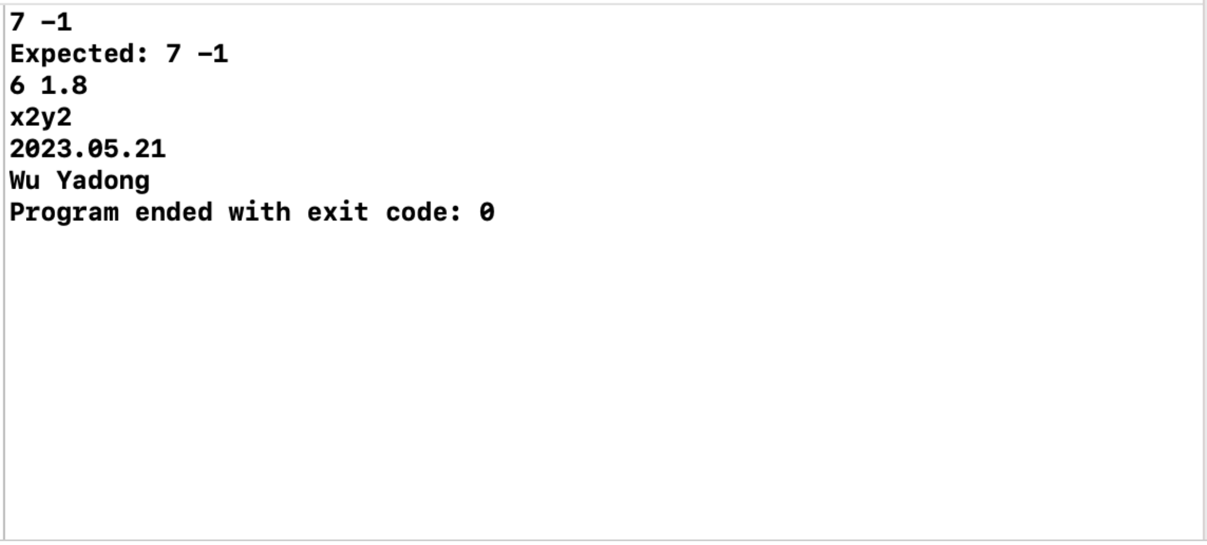
HW7P1.cpp Download

```cpp
#include <iostream>
#include "pair.h"

int main() {
    Pair<int> pair1(3, 4);
    Pair<int> result1 = pair1.sum_diff();
    std::cout << result1.get_first() << " " << result1.get_second() << std::endl;
    std::cout << "Expected: 7 -1" << std::endl;

    Pair<double> pair2(3.9, 2.1);
    Pair<double> result2 = pair2.sum_diff();
    std::cout << result2.get_first() << " " << result2.get_second() << std::endl;

    Pair<std::string> pair3("x2", "y2");
    Pair<std::string> result3 = pair3.sum_diff();
    std::cout << result3.get_first() << " " << result3.get_second() << std::endl;

    std::cout << "2023.05.21" << std::endl;
    std::cout << "Wu Yadong" << std::endl;

    return 0;
}

```

pair.h   Download

```cpp
#ifndef PAIR_H
#define PAIR_H
#include <string>
template<typename T>
class Pair {
private:
    T first;
    T second;

public:
    Pair(T first, T second) : first(first), second(second) {}

    template<typename U = T>
    Pair<typename std::enable_if<!std::is_same<U, std::string>::value, T>::type>
sum_diff() const {
        return Pair<T>(first + second, first - second);
    }

    template<typename U = T>
    Pair<typename std::enable_if<std::is_same<U, std::string>::value, std::string>::type>
sum_diff() const {
        return Pair<std::string>(first + second, "");
    }

    T get_first() const {
        return first;
    }

    T get_second() const {
        return second;
    }
};

#endif // PAIR_H

```

Screenshot 2023-05-21 at 7.47.25 PM.png Download



```
7 -1
Expected: 7 -1
6 1.8
x2y2
2023.05.21
Wu Yadong
Program ended with exit code: 0
```

Q6
14 Points

Write a recursive function `long double xpower(long double x, int n)` that calculates x^n, where $n \geq 0$. Please don't use the helper function in `xpower`.

Write a recursive function `long double xpower_h(long double x, int n)` that calculates x^n with a helper function which uses the tail recursion.

The name of the cpp file is "HW7P2.cpp".

The first part in `main()` function is given as follows

```
cout<<xpower(2,4)<<endl;
cout<<xpower(3,3)<<endl;
cout<<xpower_h(2,4)<<endl;
cout<<xpower_h(3,3)<<endl;

auto t0 = std::chrono::high_resolution_clock::now();
xpower(1.01,2000);
auto te = std::chrono::high_resolution_clock::now();
std::cout << "Without the helper function " << std::chrono::duration_cast<std::chrono::nanoseconds>(te - t0).count() <
t0 = std::chrono::high_resolution_clock::now();
xpower_h(1.01,2000);
te = std::chrono::high_resolution_clock::now();
std::cout << "With the helper function " << std::chrono::duration_cast<std::chrono::nanoseconds>(te - t0).count() << "
```

You need to add `#include <chrono>` for using `std::chrono::high_resolution_clock::now();`

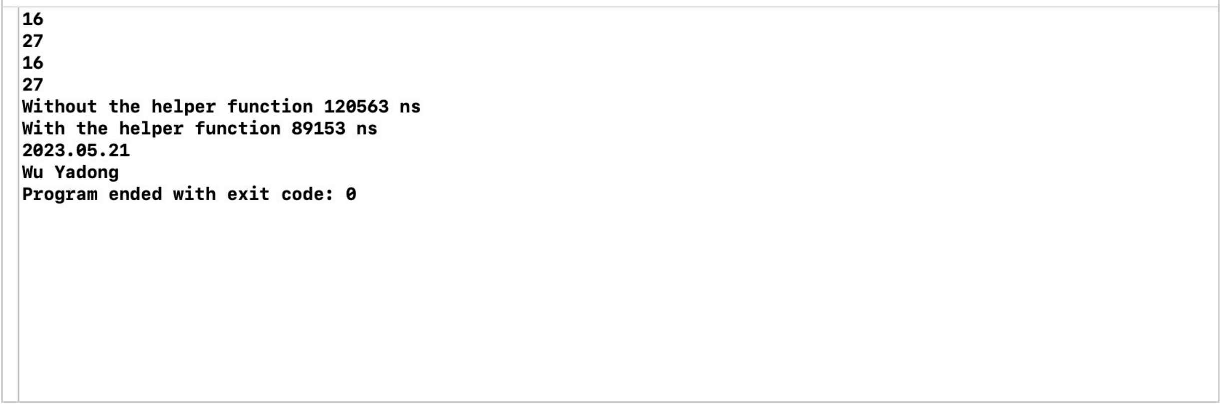
After this part, please add a `cout` statement to display the date, your name, or the name of all group members if you use group submission. Run your code and save the screenshots of the console.

Please upload "HW7P2.cpp" and screenshots here

HW7P2.cpp Download

```cpp
#include <iostream>
#include <chrono>

long double xpower(long double x, int n) {
    if (n == 0)
        return 1;
    else
        return x * xpower(x, n - 1);
}

long double xpower_h_helper(long double x, int n, long double result) {
    if (n == 0)
        return result;
    else
        return xpower_h_helper(x, n - 1, result * x);
}

long double xpower_h(long double x, int n) {
    return xpower_h_helper(x, n, 1);
}

int main() {
    std::cout << xpower(2, 4) << std::endl;
    std::cout << xpower(3, 3) << std::endl;
    std::cout << xpower_h(2, 4) << std::endl;
    std::cout << xpower_h(3, 3) << std::endl;

    auto t0 = std::chrono::high_resolution_clock::now();
    xpower(1.01, 2000);
    auto te = std::chrono::high_resolution_clock::now();
    std::cout << "Without the helper function " <<
std::chrono::duration_cast<std::chrono::nanoseconds>(te - t0).count() << " ns\n";

    t0 = std::chrono::high_resolution_clock::now();
    xpower_h(1.01, 2000);
    te = std::chrono::high_resolution_clock::now();
    std::cout << "With the helper function " <<
std::chrono::duration_cast<std::chrono::nanoseconds>(te - t0).count() << " ns\n";

    std::cout << "2023.05.21" << std::endl;
    std::cout << "Wu Yadong" << std::endl;

    return 0;
}

```

WeChata884061b8f58c001c479ac8900fce7e8.jpg Download



```
16
27
16
27
Without the helper function 120563 ns
With the helper function 89153 ns
2023.05.21
Wu Yadong
Program ended with exit code: 0
```

Remark

The XCode may not provide a good optimization for the tail recursion in the default settings. If you are an XCode user, then the time cost for these functions could be similar.