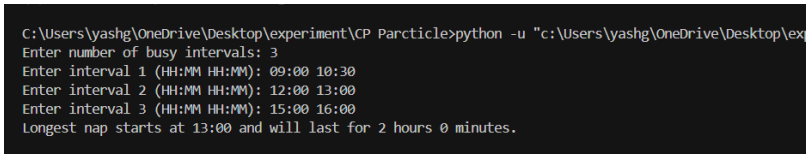
EXPERIMENT NO: 6

```
def time_to_minutes(t):
    h, m = map(int, t.split(":"))
    return h * 60 + m

def minutes_to_time(m):
    return f"{m//60:02d}:{m%60:02d}"

def longest_nap(intervals):
    WORK_START = time_to_minutes("09:00")
    WORK_END = time_to_minutes("18:00")
    intervals.append(("09:00", "09:00"))
    intervals.append(("18:00", "18:00"))

    intervals = [(time_to_minutes(s), time_to_minutes(e)) for s, e in intervals]
    intervals.sort()
    best_start, best_len = WORK_START, 0

    for i in range(len(intervals) - 1):
        gap = intervals[i+1][0] - intervals[i][1]
        if gap > best_len:
            best_len = gap
            best_start = intervals[i][1]

    return minutes_to_time(best_start), best_len

n = int(input("Enter number of busy intervals: "))
intervals = []
for i in range(n):
    s, e = input(f"Enter interval {i+1} (HH:MM HH:MM): ").split()
    intervals.append((s, e))

start, length = longest_nap(intervals)
hours, minutes = divmod(length, 60)

print(f"Longest nap starts at {start} and will last for {hours} hours {minutes} minutes.")
```




```
def round_robin(teams):
    n = len(teams)
    if n % 2 == 1:
        teams.append("BYE")
        n += 1
    rounds = n - 1
    half = n // 2
    schedule = []
    for r in range(rounds):
        matches = []
        for i in range(half):
            t1 = teams[i]
            t2 = teams[n - 1 - i]
            if t1 != "BYE" and t2 != "BYE":
                matches.append((t1, t2))
        schedule.append(matches)
        teams = [teams[0]] + [teams[-1]] + teams[1:-1]
    return schedule

n = int(input("Enter number of teams: "))
teams = []
for i in range(n):
    teams.append(input(f"Enter team {i+1} name: "))

schedule = round_robin(teams)

for i, matches in enumerate(schedule, start=1):
    print(f"\nRound {i}:")
    for m in matches:
        print(f"{m[0]} vs {m[1]}")
```


```
from collections import deque

def neighbors(word, dictionary):
    for w in dictionary:
        if sum(a != b for a, b in zip(word, w)) == 1:
            yield w

def word_ladder(start, target, dictionary):
    dictionary = set(dictionary)
    queue = deque([[start]])
    visited = {start}
    while queue:
        path = queue.popleft()
        word = path[-1]

        if word == target:
            return path

        for nxt in neighbors(word, dictionary):
            if nxt not in visited:
                visited.add(nxt)
                queue.append(path + [nxt])
    return None

start = input("Start word: ")
target = input("Target word: ")
n = int(input("Dictionary size: "))
dictionary = [input() for _ in range(n)]
res = word_ladder(start, target, dictionary)
if res:
```

```
    print(" -> ".join(res))
else:
    print("No path found.")
```

```
Start word: cat
Target word: dog
Dictionary size: 5
cat
cot
dot
cog
dog
cat -> cot -> dot -> dog
```

EXPERIMENT NO: 7

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

int countCarries(string a, string b) {
    reverse(a.begin(), a.end());
    reverse(b.begin(), b.end());
    int carry = 0;
    int carries = 0;
    int n = max(a.length(), b.length());

    for (int i = 0; i < n; i++) {
        int digitA = (i < a.length()) ? a[i] - '0' : 0;
        int digitB = (i < b.length()) ? b[i] - '0' : 0;
        int sum = digitA + digitB + carry;

        if (sum >= 10) {
            carries++;
            carry = 1;
        } else {
            carry = 0;
        }
    }
    return carries;
}

int main() {
    string a, b;
    cout << "Enter first number: ";
    cin >> a;
    cout << "Enter second number: ";
    cin >> b;
    int result = countCarries(a, b);
    cout << "Number of carry operations: " << result << endl;
    return 0;
}
```

```
Enter first number: 759
Enter second number: 846
Number of carry operations: 3
```

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

bool isPalindrome(string s) {
    string r = s;
    reverse(r.begin(), r.end());
    return s == r;
}
string addStrings(string a, string b) {
    string result = "";
    int carry = 0;
    reverse(a.begin(), a.end());
    reverse(b.begin(), b.end());
    int n = max(a.length(), b.length());
    for (int i = 0; i < n; i++) {
        int digitA = (i < a.length()) ? a[i] - '0' : 0;
        int digitB = (i < b.length()) ? b[i] - '0' : 0;
        int sum = digitA + digitB + carry;
        carry = sum / 10;
        result += (sum % 10) + '0';
    }
    if (carry > 0)
        result += (carry + '0');

    reverse(result.begin(), result.end());
    return result;
}
pair<int, string> reverseAndAdd(string num) {
    int steps = 0;
    while (steps < 1000) {
        string rev = num;
        reverse(rev.begin(), rev.end());
        num = addStrings(num, rev);
```

```
        steps++;

        if (isPalindrome(num))
            return {steps, num};
    }
    return {steps, "No palindrome found (possibly Lychrel)"};
}
int main() {
    string num;
    cout << "Enter a number: ";
    cin >> num;
    auto result = reverseAndAdd(num);
    cout << "Iterations: " << result.first << endl;
    cout << "Palindrome: " << result.second << endl;
    return 0;
}
```

```
Enter a number: 195
Iterations: 4
Palindrome: 9339
```

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

int main() {
    long long p, q;
    cout << "Enter fraction p/q: ";
    cin >> p >> q;
    long long Lp = 0, Lq = 1;
    long long Rp = 1, Rq = 0;
    string path = "";

    while (true) {
        long long mediant_p = Lp + Rp;
        long long mediant_q = Lq + Rq;
        if (mediant_p == p && mediant_q == q) {

            break;
        }
```

```
        if (p * mediant_q < q * mediant_p) {
            path += "L";
            Rp = mediant_p;
            Rq = mediant_q;
        } else {
            path += "R";
            Lp = mediant_p;
            Lq = mediant_q;
        }
    }
    cout << "Path in Stern-Brocot Tree: " << path << endl;
    return 0;
}
```

```
Enter fraction p/q: 1 5
Path in Stern-Brocot Tree: LLLL

c:\Users\yashg\OneDrive\Desktop\experiment\CP Parcticle>cd "c:\Users\yashg\One
ers\yashg\OneDrive\Desktop\experiment\CP Parcticle\"seventh_C
Enter fraction p/q: 1 2
Path in Stern-Brocot Tree: L

c:\Users\yashg\OneDrive\Desktop\experiment\CP Parcticle>cd "c:\Users\yashg\One
ers\yashg\OneDrive\Desktop\experiment\CP Parcticle\"seventh_C
Enter fraction p/q: 4 5
Path in Stern-Brocot Tree: LRRR
```

EXPERIMENT NO: 9

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

bool lessOrEqual(const string &a, const string &b) {
    if (a.size() != b.size())
        return a.size() < b.size();
    return a <= b;
}
string addBig(const string &a, const string &b) {
    int i = a.size() - 1, j = b.size() - 1, carry = 0;
    string res = "";

    while (i >= 0 || j >= 0 || carry) {
        int x = (i >= 0 ? a[i--] - '0' : 0);
        int y = (j >= 0 ? b[j--] - '0' : 0);

        int sum = x + y + carry;
        res.push_back((sum % 10) + '0');
        carry = sum / 10;
    }
    reverse(res.begin(), res.end());
    return res;
}

int main() {
    vector<string> fibs;
    fibs.push_back("1");
    fibs.push_back("2");
    while (true) {
        string next = addBig(fibs[fibs.size() - 1], fibs[fibs.size() - 2]);
        if (next.size() > 1050) break;
        fibs.push_back(next);
    }

    string A, B;
    cout << "Enter A and B (0 0 to stop):\n";
```

```
    while (cin >> A >> B) {
        if (A == "0" && B == "0")
            break;
        int count = 0;
        for (const string &f : fibs) {
            if (lessOrEqual(A, f) && lessOrEqual(f, B))
                count++;
        }
        cout << count << endl;
    }

    return 0;
}
```

```
Enter A and B (0 0 to stop):
4 20
3
2 90
9
2 200
10
0 0
```

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

int main() {
    long long n;
    cout << "Enter number of lines: ";
    cin >> n;
    long long regions = (n * (n + 1)) / 2 + 1;
    cout << "Maximum regions: " << regions << endl;
    return 0;
}
```

```
Enter number of lines: 4
Maximum regions: 11
```

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

bool isPrime(int x) {
    if (x < 2) return false;
    for (int i = 2; i * i <= x; i++)
        if (x % i == 0)
            return false;
    return true;
}
int countEven(const vector<int>& arr) {
    int count = 0;
    for (int x : arr)
        if (x % 2 == 0)
            count++;
    return count;
}
int countVowels(const string& s) {
    int count = 0;
    for (char c : s) {
        c = tolower(c);
        if (c=='a' || c=='e' || c=='i' || c=='o' || c=='u')
            count++;
    }
    return count;
}
int countPrimes(const vector<int>& arr) {
    int count = 0;
    for (int x : arr)
        if (isPrime(x))
            count++;
    return count;
}
int countCharacter(const string& s, char ch) {
    int count = 0;
    ch = tolower(ch);
    for (char c : s)
        if (tolower(c) == ch)
```

```
            count++;
    return count;
}
int countPattern(const string& s, const string& p) {
    int count = 0;
    int L = p.size();
    for (int i = 0; i + L <= (int)s.size(); i++)
        if (s.substr(i, L) == p)
            count++;
    return count;
}

int main() {
    int choice;

    cout << "Which counting do you want?\n";
    cout << "1. Count even numbers in a list\n";
    cout << "2. Count vowels in a string\n";
    cout << "3. Count prime numbers in a list\n";
    cout << "4. Count occurrences of a character in a string\n";
    cout << "5. Count occurrences of a pattern in a string\n";
    cout << "Enter your choice (1-5): ";
    cin >> choice;

    if (choice == 1) {
        int n;
        cout << "Enter list size: ";
        cin >> n;

        vector<int> arr(n);
        cout << "Enter numbers: ";
        for (int i = 0; i < n; i++)
            cin >> arr[i];
        cout << "Even count = " << countEven(arr) << endl;
    }
    else if (choice == 2) {
        cin.ignore();
        string s;
        cout << "Enter string: ";
        getline(cin, s);
```

```
    cout << "Vowel count = " << countVowels(s) << endl;
}
else if (choice == 3) {
    int n;
    cout << "Enter list size: ";
    cin >> n;

    vector<int> arr(n);
    cout << "Enter numbers: ";
    for (int i = 0; i < n; i++)
        cin >> arr[i];

    cout << "Prime count = " << countPrimes(arr) << endl;
}
else if (choice == 4) {
    cin.ignore();
    string s;
    cout << "Enter string: ";
    getline(cin, s);

    char ch;
    cout << "Enter character to count: ";
    cin >> ch;

    cout << "Occurrences of '" << ch << "' = "
         << countCharacter(s, ch) << endl;
}
else if (choice == 5) {
    cin.ignore();
    string s, pattern;
    cout << "Enter string: ";
    getline(cin, s);
    cout << "Enter pattern: ";
    cin >> pattern;
    cout << "Pattern occurs "
         << countPattern(s, pattern)
         << " times\n";
}
else {
```

```
        cout << "Invalid choice.\n";
    }
    return 0;
}
```

```
Which counting do you want?
1. Count even numbers in a list
2. Count vowels in a string
3. Count prime numbers in a list
4. Count occurrences of a character in a string
5. Count occurrences of a pattern in a string
Enter your choice (1-5): 5
Enter string: hellolahello
Enter pattern: he
Pattern occurs 2 times
```

EXPERIMENT NO: 10

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

vector<int> twoSum(const vector<int>& nums, int target) {
    int n = nums.size();

    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (nums[i] + nums[j] == target) {
                return {i, j};
            }
        }
    }
    return {};
}

int main() {
    int n, target;
    cout << "Enter size of array: ";
    cin >> n;
    vector<int> nums(n);
    cout << "Enter array elements:\n";
    for (int i = 0; i < n; i++) {
        cin >> nums[i];
    }
    cout << "Enter target sum: ";
    cin >> target;
    vector<int> ans = twoSum(nums, target);
    if (ans.empty()) {
        cout << "No pair found.\n";
    } else {
        cout << "Indices: " << ans[0] << ", " << ans[1] << endl;
    }
    return 0;
}
```

```
Enter size of array: 6
Enter array elements:
1 2 3 4 5 6
Enter target sum: 5
Indices: 0, 3
```

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

bool allUnique(const string& s, int l, int r) {
    bool seen[256] = {0};
    for (int i = l; i <= r; i++) {
        unsigned char c = s[i];
        if (seen[c])
            return false;
        seen[c] = true;
    }
    return true;
}

int main() {
    string s;
    cout << "Enter a string: ";
    cin >> s;
    int n = s.size();
    int maxLen = 0;

    for (int i = 0; i < n; i++) {
        for (int j = i; j < n; j++) {
            if (allUnique(s, i, j)) {
                int len = j - i + 1;
                if (len > maxLen)
                    maxLen = len;
            }

        }
```

```
    }
    cout << "Longest substring length (brute force): " << maxLen << endl;
    return 0;
}
```

```
Enter a string: abcdacdb
Longest substring length (brute force): 4
```