1. [Write C++ program to count number of alphabets, digits and special characters in string](https://techstudy.org/CplusplusLanguage/Write-Cplusplus-program-to-count-number-of-alphabets-digits-and-special-characters-in-string) **(10 marks)**

#include <iostream>

#include <cctype>

using namespace std;

int main() {

    string str;

    int alphabets = 0, digits = 0, special\_chars = 0;

    // Read the string

    cout << "Enter a string: ";

    getline(cin, str);

    // Iterate through each character in the string

    for (char c : str) {

        if (isalpha(c)) {

            // If the character is an alphabet, increment the alphabet count

            alphabets++;

        } else if (isdigit(c)) {

            // If the character is a digit, increment the digit count

            digits++;

        } else {

            // If the character is neither an alphabet nor a digit, it is a special character

            special\_chars++;

        }

    }

    // Print the count of alphabets, digits, and special characters

    cout << "Number of alphabets: " << alphabets << endl;

    cout << "Number of digits: " << digits << endl;

    cout << "Number of special characters: " << special\_chars << endl;

    return 0;

}

    // find\_first\_not\_of function: searches for the first occurrence of a character not in the given string in the original string

    cout << "find\_first\_not\_of('orl', 3, 3): " << str.find\_first\_not\_of("orl", 3, 3) << endl;

    // find\_last\_of function: searches for the last occurrence of any character from the given string in the original string

    cout << "find\_last\_of('orl', 5, 3): " << str.find\_last\_of("orl", 5, 3) << endl;

    // find\_last\_not\_of function: searches for the last occurrence of a character not in the given string in the original string

    cout << "find\_last\_not\_of('orl', 5): " << str.find\_last\_not\_of("orl", 5) << endl;

    // push\_back function: adds the given character to the end of the string

    str.push\_back('!');

    cout << "push\_back('!'): " << str << endl;

    // pop\_back function: removes the last character from the string

    str.pop\_back();

    cout << "pop\_back(): " << str << endl;

    return 0;

}

1. Write a C++ program to implement following string functions:

* substr(int position, int length)
* append(const string& str1)
* find(string& str1, int position, int len)
* find\_first\_of(string& str1, int position, int len)
* find\_first\_not\_of(string& str1, int position, int n)
* find\_last\_of(string& str1, int position, int n)
* find\_last\_not\_of(string& str1, int position)
* push\_back(char c)
* pop\_back()

Read the string from the user. Show the output of each function with proper message. (any 3 functions **– 10 marks**)

#include <iostream>

#include <string>

using namespace std;

int main()

{

    // Read the string from the user

    string str;

    cout << "Enter a string: ";

    getline(cin, str);

    // substr function: returns a substring of the given string

    cout << "substr(2, 5): " << str.substr(2, 5) << endl;

    // append function: appends the given string to the end of the original string

    string str1 = " World";

    str.append(str1);

    cout << "append(' World'): " << str << endl;

    // find function: searches for the given string in the original string and returns its position

    cout << "find('orld', 3, 4): " << str.find("orld", 3, 4) << endl;

    // find\_first\_of function: searches for the first occurrence of any character from the given string in the original string

    cout << "find\_first\_of('orl', 3, 3): " << str.find\_first\_of("orl", 3, 3) << endl;

    // find\_first\_not\_of function: searches for the first occurrence of a character not in the given string in the original string

    cout << "find\_first\_not\_of('orl', 3, 3): " << str.find\_first\_not\_of("orl", 3, 3) << endl;

    // find\_last\_of function: searches for the last occurrence of any character from the given string in the original string

    cout << "find\_last\_of('orl', 5, 3): " << str.find\_last\_of("orl", 5, 3) << endl;

    // find\_last\_not\_of function: searches for the last occurrence of a character not in the given string in the original string

    cout << "find\_last\_not\_of('orl', 5): " << str.find\_last\_not\_of("orl", 5) << endl;

    // push\_back function: adds the given character to the end of the string

    str.push\_back('!');

    cout << "push\_back('!'): " << str << endl;

    // pop\_back function: removes the last character from the string

    str.pop\_back();

    cout << "pop\_back(): " << str << endl;

    return 0;

}

1. Write a C++ program to calculate Compound Interest using concept of pass by reference and return by reference. Read principal amount, rate of interest and period from the user. (**10 marks**)

#include <iostream>

#include <cmath>

using namespace std;

// Function prototype

void calculateCompoundInterest(double &principal, double &rate, int &period, double &compoundInterest);

int main()

{

    double principal, rate, compoundInterest;

    int period;

    // Read principal amount, rate of interest, and period from the user

    cout << "Enter principal amount: ";

    cin >> principal;

    cout << "Enter rate of interest: ";

    cin >> rate;

    cout << "Enter period (in years): ";

    cin >> period;

    // Calculate compound interest

    calculateCompoundInterest(principal, rate, period, compoundInterest);

    // Display the compound interest

    cout << "Compound interest: " << compoundInterest << endl;

    return 0;

}

// Function to calculate compound interest

void calculateCompoundInterest(double &principal, double &rate, int &period, double &compoundInterest)

{

    compoundInterest = principal \* pow(1 + (rate / 100), period) - principal;

}

1. Write a function Interest(principal, int\_rate, year). Use function overloading and default arguments to set the values of int\_rate and year. (**10 marks**)
2. #include <iostream>
3. using namespace std;
4. double Interest(double principal, double int\_rate, int year)
5. {
6. // Calculate the interest
7. double interest = principal \* int\_rate \* year;
8. // Return the interest
9. return interest;
10. }
11. // Overloaded function with default arguments
12. double Interest(double principal, double int\_rate = 0.05)
13. {
14. int year=1;
15. // Call the original function with the default arguments
16. return Interest(principal, int\_rate, year);
17. }
18. int main()
19. {
20. // Calculate the interest on 1000 at 5% for 1 year
21. double interest1 = Interest(1000);
22. cout << "Interest: " << interest1 << endl;
23. // Calculate the interest on 1000 at 10% for 2 years
24. double interest2 = Interest(1000, 0.1, 2);
25. cout << "Interest: " << interest2 << endl;
26. return 0;
27. }
28. Create a class Account(Ano, Name, balance, int\_rate). Write a friend function Total\_bal() to calculate total interest earned on the balance. (10 marks)

#include <iostream>

#include <string>

using namespace std;

class Account

{

private:

    int A\_no;

    string Name;

    double balance;

    double int\_rate;

public:

    Account(int A, string N, double B, double IR) : A\_no(A), Name(N), balance(B), int\_rate(IR) {}

    friend double Total\_bal(const Account& a);

};

double Total\_bal(const Account& a)

{

    return a.balance \* a.int\_rate;

}

int main()

{

    Account a(12345, "John Smith", 1000.0, 0.05);

    cout << "Total interest: " << Total\_bal(a) << endl;

    return 0;

}

1. Create a class Manager(id, name, department, designation). Use default, parameterized and copy constructors to initialize the members of the class. (10 marks)

#include <iostream>

using namespace std;

class Manager

{

private:

    int id;

    string name;

    string department;

    string designation;

public:

    Manager()

        : id(0), name(""), department(""), designation("") {}

    Manager(int ID, string Name, string Department, string Designation)

        : id(ID), name(Name), department(Department), designation(Designation) {}

    Manager(const Manager& m)

        : id(m.id), name(m.name), department(m.department), designation(m.designation) {}

    void print()

    {

        cout << "ID: " << id << endl;

        cout << "Name: " << name << endl;

        cout << "Department: " << department << endl;

        cout << "Designation: " << designation << endl;

        cout << endl;

    }

};

int main()

{

    Manager m1; // Uses default constructor

    Manager m2(12345, "John Smith", "IT", "Manager"); // Uses parameterized constructor

    Manager m3(m2); // Uses copy constructor

    m1.print();

    m2.print();

    m3.print();

    return 0;

}

6 x 10 marks