set container

```
#include <iostream>
#include <set>
using namespace std;
int main()
{
set<char> a;
a.insert('G');
a.insert('F');
a.insert('G');
for (char st : a) {
cout << st << ' ';
}
cout << '\n';
return 0;
}
```

```
set<char, greater<char>> a;
```

jaccard similiarity

```
#include <list>
#include <iostream>
#include <sstream>
#include <algorithm>

using namespace std;

int main() {
    list<string> p;
    list<string> q;
    list<string> n;
    list<string> c;

    string doc1 = "Data is the new oil of digital  a economy";
    string doc2 = "Data is new oil";

    stringstream a(doc1);
    stringstream b(doc2);
    string word;

    while (a >> word) {
```

```
        p.push_back(word);
    }

    while (b >> word) {
        q.push_back(word);
    }


    p.sort();
    q.sort();

    set_intersection(
        p.begin(), p.end(),
        q.begin(), q.end(),
        back_inserter(c)
    );

    n=p;
    n.sort();
    n.unique();

    cout << "Universal:  ";
    for (string s : n) {
        cout << s << " ";
    }
    cout << endl;
    cout << "Intersection:  ";
    for (string s : c) {
        cout << s << " ";
    }
    cout << endl;

    cout << "Similarity: " << (float)c.size() / n.size() << endl;

    return 0;
}

0;
}
```

```
####### frequency of words in strings

#include <iostream>
#include <map>
```

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

int main() {
   map<string, int> doc;
   string a="learning to code is learning to code and inovate";
   stringstream p(a);
   string word;
   while(p >> word) {
         doc[word]++;
   }
   for(auto it : doc) {
      cout << it.first << ": " << it.second << endl;
   }

   return 0;
}
```

Write a program that reads oldmast.txt and trans.txt. When a match occurs (i.e., records with the same account number appear in both the master file and the transaction file), add the calculated amount in the transaction record to the current balance in the master record, and write the "newmast.txt" record.

When there is a master record for a particular account, but no corresponding transaction record, merely write the master record to "newmast.txt". When there is a transaction record, but no corresponding master record, print to a log file the message "Unmatched transaction for A/C No. ...". The log file should be a text file named "log.txt".

```
#include <iostream>
#include <map>
#include <string>
#include <sstream>
#include <fstream>
#include <iomanip>

using namespace std;

int main() {
   ifstream mFile("oldmast.txt");
```

```
ifstream tFile("trans.txt");
ofstream newMFile("newmast2.txt");
ofstream logFile("lof2.txt");

if (!mFile || !tFile || !newMFile || !logFile) {
    cerr << "Error opening files." << endl;
    return 1;
}

map<int, double> m; // Master account data
map<int, double> t; // Transaction data

string line;
int accNo;
double amount;

// Read master file using getline
while (getline(mFile, line)) {
    stringstream ss(line);
    ss >> accNo >> amount;
    m[accNo] = amount;
}

// Read transaction file using getline
while (getline(tFile, line)) {
    stringstream ss(line);
    ss >> accNo >> amount;
    t[accNo] += amount;
}

// Write updated master data
for (auto& [accNo, balance] : m) {
    if (t.count(accNo)) {
        balance += t[accNo];
        t.erase(accNo);
    }
    newMFile << accNo << " " << fixed << setprecision(2) << balance << "\n";
}

// Write unmatched transactions
for (auto& [accNo, _] : t) {
    logFile << "Unmatched transaction for A/C No. " << accNo << "\n";
}
```

```
    cout << "Processing complete. See 'newmast2.txt' and 'lof2.txt'." << endl;
    return 0;
}


######  angle
#include <iostream>
#include <cmath>
#include <vector>
using namespace std;

double dotProduct(vector<int>& A,vector<int>& B) {
    double dot = 0;
    for (int i = 0; i < A.size(); i++) {
        dot += A[i] * B[i];
    }
    return dot;
}

double magnitude(const vector<int>& V) {
    double mag = 0;
    for (int i = 0; i < V.size(); i++) {
        mag += V[i] * V[i];
    }
    return sqrt(mag);
}

int main() {
    vector<int> D1 = {1, 1, 1, 1, 0, 0};
    vector<int> D2 = {0, 0, 1, 1, 0, 1};

    double dot = dotProduct(D1, D2);
    double magD1 = magnitude(D1);
    double magD2 = magnitude(D2);

    double angle_deg = acos(dot / (magD1 * magD2))* (180.0 / 3.14);
    cout << "Angle (in degrees): " << angle_deg << "°" << endl;

    return 0;
}


##### pop if finds #
```

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

string processStack(string& str){
    stack<char>st;

    for(char ch:str){
        if(ch=='#'){
            if(!st.empty()) //geee#e#ks
                st.pop(); // geeks
        }
        else{
            st.push(ch); // gee
        }
    }

    string result;

    while(!st.empty()){
        result = st.top() + result;
        st.pop();
    }

    return result;

}

bool areEqual(string& s1,string& s2){
    return processStack(s1) == processStack(s2);
}

int main(){

    string s1 = "geee#e#ks";
    string s2;
    cout<<"Enter a string:"<<endl;
    cin>>s2;


    if(areEqual(s1,s2)){
        cout<<"Output:True"<<endl;
    }
```

```
    else{
        cout<<"Output:False"<<endl;
    }

    return 0;


}
```

Write a C program that takes two binary string as input
and estimate their simple matching coefficient (SMC)
score as follows.
binary strings

```
#include<iostream>
#include<string.h>

using namespace std;

int main(){
    char x[20];
    cout<<"input a binary \n";
    fgets(x, sizeof x, stdin);
    char y[20];
    cout<<"input a binary\n";
    fgets(y, sizeof y, stdin);
    int f01=0, f10=0,f00=0,f11=0;

    for(int i=0; i<strlen(x); i++){
        for(int j=0; j<strlen(y); j++){
            if(x[i]=='0' && y[j]=='1'){
                f01++;
            }
            else if(x[i]=='1' && y[j]=='0'){
                f10++;
            }
            else if(x[i]=='0' && y[j]=='0'){
                f00++;
            }
            else{
                f11++;
            }
```

```
      }
   }
   cout<<f00<<endl;
   cout<<f11<<endl;
   cout<<f10<<endl;
   cout<<f01<<endl;
   double SMC=(double)(f11+f00)/(f01+f10+f11+f00);
   cout<< SMC ;
   return 0;


}
```

setA

1. You were dreaming of a journey to the planet “RetroP” on a spaceship in space represented as a
plane. You are going to start your journey at point (x, y) and the planet is located at $(P\,x, P\,y)$ point.
The navigation system of your spaceship follows a list of orders which can be represented as a string S. The system reads S from left to right where each letter represents a direction towards planet “Retrop”.

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

void location(int x, int y, string& direction){
   for(int i=0; i<=direction.size();i++){  // direction.length()    direction(i) != '\0'
      switch(direction[i]){
         case 'R':
            x++;
            break;

         case 'L':
            x--;
            break;

         case 'U':
            y++;
            break;

         case 'D':
            y--;
            break;
```

```
        }
    }
    cout << "your final location" << x << "," << y<< endl;
}

int main(){
    int x,y;
    string direction;
    cout << "enter initial location" << endl;
    cin >> x >> y;
    getchar();
    cin >> direction;
    location(x,y,direction);

}
```

2. In recent days, applicants had to apply for admission in CU using SMS. The format for the SMS
was “CU <First 3 letters of HSC education board> <HSC Roll> <HSC Passing Year> <First 3 letters of SSC education board> <SSC Roll> <SSC Passing Year>”. Each part of the SMS will be
separated by a space.

```
#include <iostream>
#include <cctype>
#include <sstream>

using namespace std;

int validate(string& sms){
    string cu, division_ssc, roll_ssc, division_hsc, roll_hsc;
    int ssc_year, hsc_year;
    stringstream ss(sms);
    ss >> cu >> division_ssc >> roll_ssc >> ssc_year  >> division_hsc >> roll_hsc >> hsc_year;
    if (cu!="CU") return 0;
    if (division_hsc.size()!=3||division_ssc.size()!=3) return 0;
    for(int i=0; i<=2; i++){
        if(!isupper(division_hsc[i]) || !isupper(division_ssc[i])) return 0;
    }
    if (ssc_year+2>hsc_year) return 0;

    return 1;
}
```

```
int main(){
    string sms;
    cout<< "enter your sms"<< endl;
    getline(cin, sms);
    cout << validate(sms);
    return 0;
}
```

3. ############You are given a text file named “numbers.txt” which contains some numbers in each line that are
separated with commas. Write a Java program to read the file and for each line print the max of the numbers in console

```
#include <iostream>
#include <set>
#include <sstream>
#include <fstream>
using namespace std;
int main(){
    set< int, greater<int>> p;
    ifstream file("1.txt");
    ofstream file2("2.txt");
    string numbers;
    while(getline(file, numbers )){
        for(int i=0; i<numbers.size(); i++){
            if (numbers[i]==',')
                numbers[i]=' ';
        }
        stringstream ss(numbers);
        int num;
        while(ss >> num){
            p.emplace(num);
        }
        for(auto it: p){
            cout<<it<< endl;
            file2 << it << "\n";
            break;
        }
        p.clear();
    }

}
```

4. ###### Suppose a balloon seller is giving you a balloon every minute. Each balloon has a color, a serial
number, and a size. You can receive multiple ballons of the same color.
Suppose you want to keep track of how many different colors of balloon you have using a C++ program. Which STL interface are you going to use? Write the necessary code for the task.

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

int main(){
    map<string, int>p;
    while(1){
        string color;
        cout<<"enter color: ";
        cin >> color;
        if(color=="q") break;
        else{
            p[color]++;
        }

    }
    for (auto it: p){
        cout << it.first << "  " << it.second << endl;
    }
}
```

5. ###### Hash tables are auxiliary data structures that map indexes to keys. However, hashing these keys
may result in collisions, meaning different keys generate the same index in the hash table. Linear
probing is one of many algorithms designed to find the correct position of a key in a hash table. In linear probing, the hash table is searched sequentially that starts from the original location of the hash.

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

int main(){
    int hash[7];
    int index;
    for(int j=0;j<7;j++){
```

```
        hash[j] = -1;
    }
    int keys[]={76,93,40,47,10,55,11};
    for(int i=0;i<7;i++){
        index = keys[i] % 7;
        while(hash[index]!=-1){
            index=(index+1) % 7;
        }
        hash[index] = keys[i];
    }
    for(auto it:hash){
        cout<<it<<endl;
    }
}
```

#####set2

1. In mathematics, a semi-prime is a natural number that is the product of two prime numbers. The
semi-primes less than 100 are:
4, 6, 9, 10, 14, 15, 21, 22, 25, 26, 33, 34, 35, 38, 39, 46, 49, 51, 55, 57, 58, 62, 65, 69, 74, 77, 82,
85, 86, 87, 91, 93, 94, and 95
Write down a program that will take an integer N as input and will determine whether N is semi prime or not.

```
#include <iostream>
using namespace std;
int is_prime(int num) {
    if(num<=1) return 0;
    if(num<=3) return 1;
    for(int i=3; i*i<=num; i++) {
        if(num%i==0) return 0;
    }
    return 1;
}
int is_semi_prime(int num) {
    for(int i=2; i<=num; i++) {
        if(is_prime(i)) {
            for(int j; j<=num; j++) {
                if(is_prime(j)) {
                    if(i*j==num) return 1;
```

```
            }
        }
    }
}
}

int main() {
    cout << "enter a num";
    int num;
    cin>> num;
    if(is_semi_prime(num)) cout<< "semi prime";
    else cout << "not semi prime";
}
```

2. Write a user-defined function to check the strength of a password for a registration system. The
function returns 1 if the pasgword is strong; otherwise, it returns -1, adhering to specific criteria for a legal password as follows:

- The password needs to have at least 12 characters;
- The password has to be a mix of lowercase and uppercase letters;
- The password must contain at least one number;
- The password must contain at least any one of these special characters) , . ; : <> (

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

int main(){
    string pass;
    cout << "enter pass";
    getline(cin, pass);
    cout<< "your pass is:" << pass<< endl;
    int count_lower=0, count_upper=0, count_num=0, count_special=0;
    for(int i=0; i<pass.size();i++){
        if(isupper(pass[i])) count_upper++;
        else if(islower(pass[i])) count_lower++;
        else if(isdigit(pass[i])) count_num++;
        else count_special++;
    }
    if(pass.size()==12 && count_num>=1 && count_lower>=1 && count_upper>=1 &&
count_special>=1) cout << "1";
    else cout << "-1";
}
```

3. A unique student ID is assigned to each student when they register for a program. The digits of a
student's ID represent various information. For example: first 2 digits indicate the batch number the fourth and fifth digits combinedly indicate the department (01-MPE, 02-EEE, 03-WEI 04-CSEI 05-CSE, 06-BTM), and the third digit from the last indicates the section.

```
#include <iostream>
using namespace std;
void id_check(int id) {
    int batch= id/10000000;
    cout << "Batch: " << batch << endl;
    int dept= (id/10000)%100;
    switch (dept) {
    case 2:
        cout << "Dept: MPE"<< endl;
        break;
    case 3:
        cout << "Dept: EEE"<< endl;
        break;
    case 4:
        cout << "Dept: WEI"<< endl;
        break;
    }
    int section= (id/100)%10;
    cout << "Section: "<< section;

}
int main() {
    int id;
    cout << "enter your id";
    cin>> id;
    id_check(id);

}
```

4. There is a le bookData.txt that contains the data for the number of new books to be bought for
different courses in a university. The le contains several lines in the format: CourseName-NumberOfBooks. Write a program that will read the file. Then it will find out the course which will need the maximum number of books and write it in similar format in an output file called max.txt.

```
#include<iostream>
#include<sstream>
```

```
#include<fstream>
using namespace std;
int main(){
   ifstream file("4.txt");
   string books,max_book="";
   int maxbook_count=0;
   while(getline(file,books )){
      for(int i=0; i<books.size(); i++){
         if(books[i]=='-') books[i]=' ';
      }
      stringstream ss(books);
      string book;
      int num=0;
      ss >> book >> num;
      if(num>maxbook_count){
         maxbook_count=num;
         max_book= book;
      }
   }
   ofstream file1("out.txt");
   file1 << max_book<< "-" << maxbook_count;
}
```

5. Jaro Similarity is the measure of similarity between two strings. The value of Jaro distance ranges
from 0 to 1. where 1 means the strings are equal and 0 means no similarity between the two strings.
The Jaro Similarity is calculated using the following formula:
where:
m is the number of matching characters;
|s1| and |s2| are the lengths of strings s1 and s2, respectively;
t is half the number of transpositions i.e. half the number of matching characters in both strings but in a different order.

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

int main() {
   string doc1 = "WINKLER";
   string doc2 = "WELFARE";
   int s1_l = doc1.size(), s2_l = doc2.size();
```

```
    int match_distance = max(s1_l, s2_l) / 2 - 1;
    vector<bool> doc1_match(s1_l, false);
    vector<bool> doc2_match(s2_l, false);
    int m = 0;

    // Find matching characters
    for(int i = 0; i < s1_l; i++) {
        int start = max(0, i - match_distance);
        int end = min(i + match_distance + 1, s2_l);
        for(int j = start; j < end; j++) {
            if(!doc2_match[j] && doc1[i] == doc2[j]) {
                doc1_match[i] = true;
                doc2_match[j] = true;
                m++;
                break;
            }
        }
    }

    // Count transpositions
    int t = 0, k = 0;
    for(int i = 0; i < s1_l; i++) {
        if(doc1_match[i]) {
            while(!doc2_match[k]) k++;
            if(doc1[i] != doc2[k]) t++;
            k++;
        }
    }
    t /= 2;

    double jaro = 0.0;
    if(m != 0) {
        jaro = (1.0 / 3) * ((double)m / s1_l + (double)m / s2_l + (double)(m - t) / m);
    }

    cout << "Matching characters (m): " << m << endl;
    cout << "Transpositions (t): " << t << endl;
    cout << "Jaro Similarity: " << jaro << endl;

    return 0;
}
```

chikungunia

```
#include <iostream>
#include <map>
#include <fstream>
#include <sstream>
#include <string>

using namespace std;

int main(){
    ifstream file1("1.txt");
    string line;
    map<string, int>location_count;

    while(getline(file1, line)){
        for(char& it : line){
            if(it=='-') it= ' ';
        }
        stringstream ss(line);
        string name, location;
        int age;
        ss>> name >> age >> location;
        location_count[location]+= 1;

    }
    string max_dis=" ";
    int max_count= 0;
    for(auto it : location_count){
        if(it.second> max_count){
            max_count= it.second;
            max_dis= it.first;
        }
    }
    cout << max_dis << "-" << max_count;
    ofstream file2("2.txt");
    file2 << max_dis << "-" << max_count;

    file1.close();
    file2.close();
}
```

salary ticket

```
#include <iostream>
#include <map>
#include <set>
#include <stdlib.h>
#include <sstream>

using namespace std;

int main() {
    map<string, int> salary_ticket;
    set<string> travelers;
    FILE *file = fopen("salary_ticket.txt", "r");
    FILE *file1 = fopen("travelers.txt", "r");
    FILE *file2 = fopen("salary_ticket_price.txt", "w");
    if (file == NULL) {
        cerr << "Error opening file" << endl;
        return 1;
    }
    char buffer[1024];
    while (fgets(buffer, sizeof(buffer), file) != NULL) {
        for(int i = 0; i < strlen(buffer); i++) {
            if (buffer[i] == ' ') {
                buffer[i] = '\n';
            }
            if(buffer[i] == ':') {
                buffer[i] = ' ';
            }
        }
        string line(buffer);
        stringstream ss(line);
        string name;
        int salary;
        ss >> name >> salary;
        salary_ticket[name] = salary;
    }
    cout << "Salary Ticket\n";
    for(auto it : salary_ticket) {
        cout << it.first << "  " << it.second << endl;
    }
    while(fgets(buffer, sizeof(buffer), file1)){
        string line(buffer);
        stringstream ss(line);
```

```
        string name;
        ss >> name;
        travelers.insert(name);
    }
    cout << "Travelers\n";
    for(auto it : travelers) {
        cout << it << endl;
    }
    for(auto it : travelers) {
        float price = salary_ticket[it] * 0.1;
        fprintf(file2, "%s %.2f\n", it.c_str(), price);
    }

    fclose(file);
    return 0;
}
```

mastercard

```
#include <iostream>
#include <map>
#include <sstream>

using namespace std;

int main() {
    FILE *oldfile= fopen("oldfile.txt","r");
    FILE *transfile= fopen("transfile.txt","r");
    FILE *newfile= fopen("newfile.txt","w");
    FILE *logfile= fopen("logfile.txt","w");

    char buffer[100];

    map <int, double> old_balance;
    map <int, double> trans_balance;
    map <int, double> new_balance;

    int id;
    double amount;

    while(fgets(buffer, sizeof(buffer), oldfile)) {
        string line(buffer);
        stringstream ss(line);
```

```
        ss >> id >> amount;
        old_balance[id] = amount;
    }

    while(fgets(buffer, sizeof(buffer), transfile)) {
        stringstream ss(buffer);
        ss >> id >> amount;
        trans_balance[id] = amount;
    }

    for(auto it : old_balance) {

        cout << "Processing account " << it.first << " " << it.second << endl;

        id = it.first;
        amount = it.second;

        double n_balance = amount + trans_balance[id];
        new_balance[id] = n_balance;

        cout << "New balance for account " << id << "  " << n_balance << endl;
        fprintf(newfile, "%d %f\n", id, n_balance);

        cout << "Transaction details for account " << id << "  " << amount << "  " <<
trans_balance[id] << "  " << n_balance << endl;
        fprintf(logfile, "%d %f %f %f ", id, amount, trans_balance[id], n_balance);

    }

    fclose(oldfile);
    fclose(transfile);
    fclose(newfile);
    fclose(logfile);

    return 0;
}

######## prime in range

#include<stdio.h>
int prime(int start){
    if(start<=1){
        return 0;
    }
```

```
    for(int i= 2; i*i <= start ; i++){
        if(start%i==0)
            {
                return 0;
            }

    }
    return 1;
}
int prime_range(int start , int end)
{
    if(start>end){
        printf("invalid input");
        return 0;
    }
    for(int i=start; i<=end; i++){
         if(prime(i)){
             printf("%d \t", i);
         }
     }
    printf("\n");
    return 1;
}
int main()
{
    int start, end;
    printf("start:");
    scanf("%d", &start);

    printf("end:");
    scanf("%d", &end );

    printf("prime numbers:");
    prime_range(start, end);

}

###### sstream

#include<iostream>
#include <sstream>
using namespace std;
 int main(){
     string s = "Dept. of cse CU";
```

```
    stringstream ss(s);
    string word;
    while(ss>> word){
       cout<< word << endl;
    }
    cout << endl;
    return 0;
}
```

character count

```
#include <iostream>
   #include <sstream>
  #include <stack>
  #include <map>
  #include <cstring>

 using namespace std;

int main() {
   map<string, int> charCount;
  FILE *file
   if (!file) {
        cout << "File couldn't be opened." << endl;
         return 1;
     }

     char buffer[1000];
     while (fgets(buffer, sizeof(buffer), file) != NULL) {
        string str(buffer);
        for (int i = 0; i < str.length(); i++) {
           string ch = str.substr(i, 1);
           if (charCount.empty()) {
                charCount[ch] = 1;
           } else {
               if (charCount.find(ch) == charCount.end()) {
                   charCount[ch] = 1;
                } else {
                   charCount[ch]++;
}
}
}
fclose(file);
```

letter manupulation

```
#include<iostream>
#include<sstream>
#include<cstring>

using namespace std;

int main(){

FILE *f1=fopen("in.txt","r");
FILE *f2=fopen("ou.txt","w");
char ch[100];
string result=" ";

if(!f1 || !f2){
  cout<<"error opening file\n";
return 1;
}

while((fgets(ch,sizeof(ch),f1))){

   stringstream ss(ch);
   string word;
    ss>>word;


     for(int i=0;i<word.length();i++){
      result += word[i] + 4;



  cout<<"Shifted character:"<<endl;
  fprintf(f2,"%s",result.c_str());
     }
  }
fclose(f1);
fclose(f2);

return 0;
}
```