Prime_Checker Algorithm

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


int power(long long a, long long n, long long p)
{
    long long int result = 1;
    a = a % p;
    while(n > 0)
    {
        if(n & 1)
            result = (result * a) % p;
        n = n >> 1;
        a = (a*a)% p;
    }
    return result;
}

int gcd(long long int r1, long long int r2)
{
    int remainder;
    int quotient;
    while(r2 != 0)
    {
        remainder = r1 % r2;
        quotient = r1 / r2;
        r1 = r2;
        r2 = remainder;
    }
    return r1;
}
bool isPrime(long long num)
{
    if(num <= 1 || num == 4)
        return false;
    if(num <= 3)
        return true;
    int k = 1000;
    while( k > 0)
    {
        long long a = 2 + rand() % (num - 4);
        if(gcd(num,a) != 1)
            return false;
        if(power(a, num-1, num) != 1)
            return false;
        k--;
    }
    return true;
}
int main()
{
    long long number;
    cout << "Enter number for checking prime :";
    cin >> number;
    bool iS_prime = isPrime(number);
    if(iS_prime)
        cout << "number is a prime !";
    else
        cout <<"Number is not a prime";

}
```

Randomized_mvc Lab : 11

```
#include<bits/stdc++.h>
using namespace std;

vector<pair<int, int>> removerdge(vector<pair<int, int>> graph,int rand_choose)
{
     int u = graph[rand_choose].first, v= graph[rand_choose].second;
     vector<pair<int, int>> newedges;
     int total_edge = graph.size();
     for(int i=0; i<total_edge; i++)
        {
            if(graph[i].first != u && graph[i].second != u && graph[i].second != v && graph[i].first != v)
            {
                newedges.push_back(make_pair(graph[i].first,graph[i].second));
            }
        }
        return newedges;

}

int main()
{
    int vertices, edges;
    cin >> vertices >> edges;
    vector<pair<int, int>> graph(edges);

    vector<int> Canswer;
    for(int i=0; i<edges; i++)
    {
        int u,v;
        cin >> u >> v;
        graph.push_back(make_pair(u,v));
    }
    // cout << graph.size() << " ";
    while(graph.size() > 0)
    {
        int total_edge = graph.size();
        int rand_choose = rand() % total_edge;
        int u = graph[rand_choose].first, v= graph[rand_choose].second;
        Canswer.push_back(u);
        Canswer.push_back(v);
        graph = removerdge(graph, rand_choose);
    }
    for(auto val: Canswer)
    {
        cout << val << " ";
    }

    return 0;
}
```

Lab 10 : Load Balancing Randomized

```
// answer should be within some range
// easy to solve

#include<bits/stdc++.h>
using namespace std;

int main()
{
    srand(time(0));
    int totaljobs = 1000;
    int jobs[totaljobs];
    int total_load = 0;
    for(int i=0; i<totaljobs; i++)
      {
          jobs[i] = rand() % 83;
          total_load += jobs[i];
      }
    cout <<"how many machines ? "<< endl;
    int machines;
    cin >> machines;
    cout << "expected load on each machines :" << endl;
    int average = total_load / machines ;
    cout << total_load / machines << "\n";

    int machine_load[machines];
    for(int i=0; i<machines; i++)
        machine_load[i] = 0;
   vector<vector<int>> machine_job(machines);

    for(int i=0; i<totaljobs; i++)
    {
        int index = rand() % machines;
        machine_job[index].push_back(i);
        machine_load[index] += jobs[i];
    }
    int min = 999999;
    int max = 0;
    for(int i=0; i<machines; i++)
    {
      if(machine_load[i] > max)
      max = machine_load[i];
      if(machine_load[i] <min)
      min = machine_load[i];
    }
    cout << "maximum load: "<< max << endl;
    cout << "Minimum Load: "<< min << endl;
    int sum = 0;
    for(int i=0; i<machines; i++)
    {
        int temp = machine_load[i] - average;
        sum += pow(temp, 2);
        cout << machine_load[i] << " ";
    }
    int ans = sum / machines;
    float ans2 = sqrt(ans);

    cout << "\nStandatd deviation : "<< ans2 << endl;
    // for(int i=0; i<machines; i++)
    // {
    // cout << "\njob no: ";
    // for(int j=0; j<machine_job[i].size(); j++)
    // {

    // cout << machine_job[i][j] <<" ";
    // }
    // }

    return 0;
}
```

LAB 8 : Reduction SOS to KS (16/ 09/ 2022)

// SOS can be reduced to KS, but reverse is not true.

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int total_item, sum;
    cout << "Enter a size of set : \n";
    cin >> total_item;
    cout << "Enter a set element one by one\n" ;
    vector<int> weight(total_item);
    for(int i=0; i<total_item; i++)
        cin >> weight[i];
    cout << "Enter Sum of subset that you want" << endl;
    cin >> sum;

    int arr[total_item+1][sum+1];
    for(int i=0; i<sum+1; i++)
        arr[0][i] = 0;

    for(int i=0; i<total_item+1; i++)
        arr[i][0] = 0;

    for(int i=1; i<total_item+1; i++)
    {
        for(int w=1; w<sum+1; w++)
        {
            if(w < weight[i])
                arr[i][w] = arr[i-1][w];
            else
            {
                if(weight[i]+arr[i-1][w-weight[i]] >= arr[i-1][w])
                {
                    arr[i][w] = weight[i]+arr[i-1][w-weight[i]];
                }
                else
                {

                    arr[i][w] = arr[i-1][w];
                }

            }

        }
    }
    if(arr[total_item][sum] == sum)
    {
        cout << "YES " << endl;
        int i = total_item;
        int j = sum;
        int taken[total_item];
        memset(taken, 0, 4);
        while(i>0 && j > 0)
        {
            if(arr[i][j] == arr[i-1][j])
            {
                taken[i] = 0;
                i--;
            }
            else
            {
                taken[i] = 1;
                j= j- weight[i];
                i--;
            }

        }
        for(int i=0; i<total_item; i++)
        {
            if(taken[i])
                cout << weight[i] << " ";
        }
    }
    else
    {
        cout << "NO" << endl;
    }

    return 0;
}

/*
[user1@hadoop-clone-42 AA]$ ./a.out
Enter a size of set :
4
Enter a set element one by one
2
3
5
6
Enter Sum of subset that you want
8
YES
3 5
*/
```

Intersectopn LAB 9: (14/09/2022)

```
#include<bits/stdc++.h>
using namespace std;
typedef struct Points{
    int x = 0;
    int y = 0;
}point;

int direction(point pi, point pj, point pk)
{
    int x1 = (pk.x - pi.x);
    int y1 = (pk.y - pi.y);
    int x2 = (pj.x - pi.x);
    int y2 = (pj.y - pi.y);
    return ((x1*y2) - (x2*y1));
}
bool onsegment(point pi, point pj, point pk)
{
    if(pk.x >= min(pj.x,pi.x) && pk.x <= max(pi.x,pj.x) && pk.y <= max(pi.y, pj.y) && pk.y >=
min(pi.y, pj.y))
        return true;
    return false;
}

int main()
{
    point p1,p2,p3,p4;
    cout << "First line segment x-y coordinate : \n" ;
    cin >> p1.x >> p1.y >> p2.x >> p2.y;
     cout << "second line segment x-y coordinate : \n" ;
    cin >> p3.x >> p3.y >> p4.x >> p4.y;
    int d1 = direction(p3, p4, p1);
    int d2 = direction(p3, p4, p2);
    int d3 = direction(p1, p2, p3);
    int d4 = direction(p1, p2, p4);

    if(d1*d2 < 0 && d3*d4 < 0)
        cout << "Yes intesects 1 "<<endl;
    else if(d1 == 0 && onsegment(p3, p4, p1))
         cout << "Yes intesects 2 "<<endl;
    else if(d2 == 0 && onsegment(p3, p4, p2))
         cout << "Yes intesects 3 "<<endl;
    else if(d3 == 0 && onsegment(p1, p2, p3))
         cout << "Yes intesects 4 "<<endl;
    else if(d4 == 0 && onsegment(p1, p2, p4))
         cout << "Yes intesects 5 "<<endl;
    else
    {
        cout << "NO intersection" << endl;
    }

    return 0;
}
```

automata

```
#include <iostream>
#include <bits/stdc++.h>
#include <string.h>
using namespace std;
bool is_not_suffix(string pattern, int k, int symb)
{
     str ch = symb == 0 ? 'a' : 'b';
     int temp = pattern.length() - k -1;
     string str = pattern.substr(temp, k);
     cout << str << endl;
}
int main()
{
    string text = "JIMY_HAILED_THE_LEADER_TO_STOP";
    string pattern = "LEADER";
    int n = text.length();
    int m = pattern.length();
    int arr[m+1][2] = {0};
    for(int q = 0; q<m ;q++)
    {
        for(int symb = 0; symb < 2; symb++)
        {
            int k = min(n, q+1);
            bool is_not_suffix = find(pattern, k, symb);
            while(is_not_suffix)
            {
                k--;
            }

        }
    }
    return 0;
}
```

hORSEPOOL

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;
```

```
int main()
{
    string text = "JIMY_HAILED_THE_LEADER_TO_STOP";
    string pattern = "LEADER";
    int n = text.length();
    int m = pattern.length();
    unordered_map<char, int> mp;
    for(int i=0; i<26; i++)
    {
        mp[65+i] = m;
    }
    mp[95] = m;
    for(int i=0; i<m-1; i++)
    {
        mp[pattern[i]] = m - i -1;
    }
    cout << m <<" " << n << endl;
    int j =0;
    while(j+m <= n)
    {
        if(pattern[m-1] == text[j+m-1])
        {
            int i = m-2;
            while( i>= 0 && pattern[i] == text[j+i])
            {
                i--;
            }
            if(i==-1)
                cout << "match at : " << j <<endl;
        }
        j = j + mp[text[j+m-1]];
    }

}
```

Naive Stringmatch

```
// implement that in coreman exercise
#include<iostream>
using namespace std;
int main()
{
    string str1 = "ababaabcaabaab";
    string find = "aab";
    int n = str1.length();
    int m = find.length();
    int shift = 0;
    for(int shift = 0; shift <= n-m ; shift++)
    {
        bool flag = true;
        for(int j=0; j<m; j++)
        {
            if(str1[shift+j] != find[j])
            {
                flag = false;
                break;
            }
        }
        if(flag)
        {
            cout << "occurrence is at : " << shift << endl;
        }
    }

}
```

Randomized find

```
#include <bits/stdc++.h>
using namespace std;
int find(vector<int> arr, int smallest)
{
    int y = rand() % (arr.size());
    int pivot = arr[y];
    vector<int> seta, setb;
    for(int i=0; i<arr.size(); i++)
    {
        if(arr[i] < pivot)
            seta.push_back(arr[i]);
        else
            setb.push_back(arr[i]);
    }
    int asize = seta.size();
    int bsize = setb.size();
    int answer;
    if(asize == smallest-1)
    {
        return pivot;
    }
    else if(asize < smallest-1 )
    {
        answer = find(setb, smallest - (asize ));
    }
    else
    {
        answer = find(seta, smallest);
    }
    return answer;
}
int main()
{
    vector<int> arr = {2,8,3,9,7,16,23,4};
    int smallest = 5;
    int answer = find(arr, smallest);
    cout << answer << " is answer" ;
}
```

Kargers Algorithm

```
#include<bits/stdc++.h>
#include<iostream>
using namespace std;

int main()
{
    int vertices = 4;
    int graph[vertices][vertices] = {
        {0,1,1,1},
        {1,0,0,1},
        {1,0,0,1},
        {1,1,1,0}
    };


    int n = 1;
    while(n > 0)
    {
    int u,v;
    cout << "Enter edge vertices :";
    cin >> u >> v;
    set<int> Set;

     int g1[vertices][vertices];

     for(int i=0; i<vertices; i++)
        for(int j=0; j<vertices; j++)
            g1[i][j] = graph[i][j];

    for(int i=0; i<vertices-1; i++)
    {
        if(Set.find(i) != Set.end())
        {
            int l;
            for(auto it = Set.begin(); it != Set.end(); it++)
        {
            if(*it < i)
            {
                l = *it;
                break;
            }
            for(int m=0; m< vertices; m++)
            {
                g1[i][m] = g1[l][m];
            }
            continue;
        }

    }
    for(int j=i+1; j <vertices; j++)
    {
        int count = 0;
        if(i==u || j==v)
        {
            g1[i][j] = 0;
            g1[j][i] = 0;
        }
        else if(i == u || i == v)
        {
            if(g1[u][j] > 0)
                count += g1[u][j];
            if(g1[v][j] > 0)
                count += g1[v][j];
            g1[u][j] = count;
            g1[v][j] = count;
        }
        else if(j == u || j == v)
        {
            if(g1[u][i] > 0)
                count += g1[u][i];
            if(g1[v][i] > 0)
                count += g1[v][i];
            g1[u][i] = count;
            g1[v][i] = count;
        }
    }
    cout << endl;
for(int ii=0 ;ii<vertices; ii++)
{
    for(int jj=0; jj<vertices; jj++)
    {
        cout << g1[ii][jj] <<" ";
    }
```

```
        cout << endl;
    }
        Set.insert(u);
        Set.insert(v);
        }


    vertices--;
    int g2[vertices][vertices];
    int x = 0, y = 0;
    for(int i=0; i< vertices; i++)
    {
        if(i == v)
            continue;
        for(int j=0; j<vertices; j++)
        {
            if(j == v) continue;
            else
            {
                g2[x][y] = g1[i][j];
                y++;
            }

        }
        x++;
    }
    for(int i=0 ;i<vertices; i++)
        for(int j=0; j<vertices; j++)
        {
            graph[i][j] = g2[i][j];
        }
        n--;
}


    return 0;
}
```

Lab 10 : Load Balancing Greedy

```
// answer should be within some range
// easy to solve

#include<bits/stdc++.h>
using namespace std;

int main()
{
    int totaljobs;
    cin >> totaljobs;
    int jobs[totaljobs];
    for(int i=0; i<totaljobs; i++)
        cin >> jobs[i];

    cout <<"how many machines ? "<< endl;
    int machines;
    cin >> machines;
    int machine_load[machines];
    for(int i=0; i<machines; i++)
        machine_load[i] = 0;
   vector<vector<int>> machine_job(machines);

    for(int i=0; i<totaljobs; i++)
    {
        int min_load_index = 9999;
        int index;
        for(int j=0; j<machines; j++)
        {
            if(machine_load[j] < min_load_index)
            {
                min_load_index = machine_load[j];
                index = j;
            }

        }
        machine_job[index].push_back(i);
        machine_load[index] += jobs[i];
    }
    for(int i=0; i<machines; i++)
    {
        cout << machine_load[i] << " ";
    }
    for(int i=0; i<machines; i++)
    {
        cout << "\njob no: ";
        for(int j=0; j<machine_job[i].size(); j++)
            {

                cout << machine_job[i][j] <<" ";
            }
    }

    return 0;
}
```