# CAPSTON – 3

## Task - 1

```
#include <iostream>
using namespace std;
#include <sys/sysinfo.h>
void displayMemoryInfo(){
	struct sysinfo info;
	if (sysinfo(&info)==0){
		cout<< "Total used RAM (info.totalram in MB): "<<info.totalram/(1024*1024)<<"MB\n";
		cout<< "Total used RAM (info.totalram in GB): "<<(info.totalram/(1024*1024))/1024<<"GB\n";
		cout<< "Total unused RAM (info.freeram in MB): "<<info.freeram/(1024*1024)<<"MB\n";
		cout<< "Total shared RAM (info.sharedram in MB): "<<info.sharedram/(1024*1024)<<"MB\n";
	}
}
int main(){
	displayMemoryInfo();
	return 0;
}
```

```
anil_pavilion@Anil-Pavilion: ~/2141007073/system_monitor_tool
anil_pavilion@Anil-Pavilion:~/2141007073/system_monitor_tool$ g++ --std=c++17 task1.cpp -o t1
anil_pavilion@Anil-Pavilion:~/2141007073/system_monitor_tool$ ./t1
Total used RAM (info.totalram in MB): 3651MB
Total used RAM (info.totalram in GB): 3GB
Total unused RAM (info.freeram in MB): 2725MB
Total shared RAM (info.sharedram in MB): 3MB
anil_pavilion@Anil-Pavilion:~/2141007073/system_monitor_tool$
```

# Task - 2

## Part 1

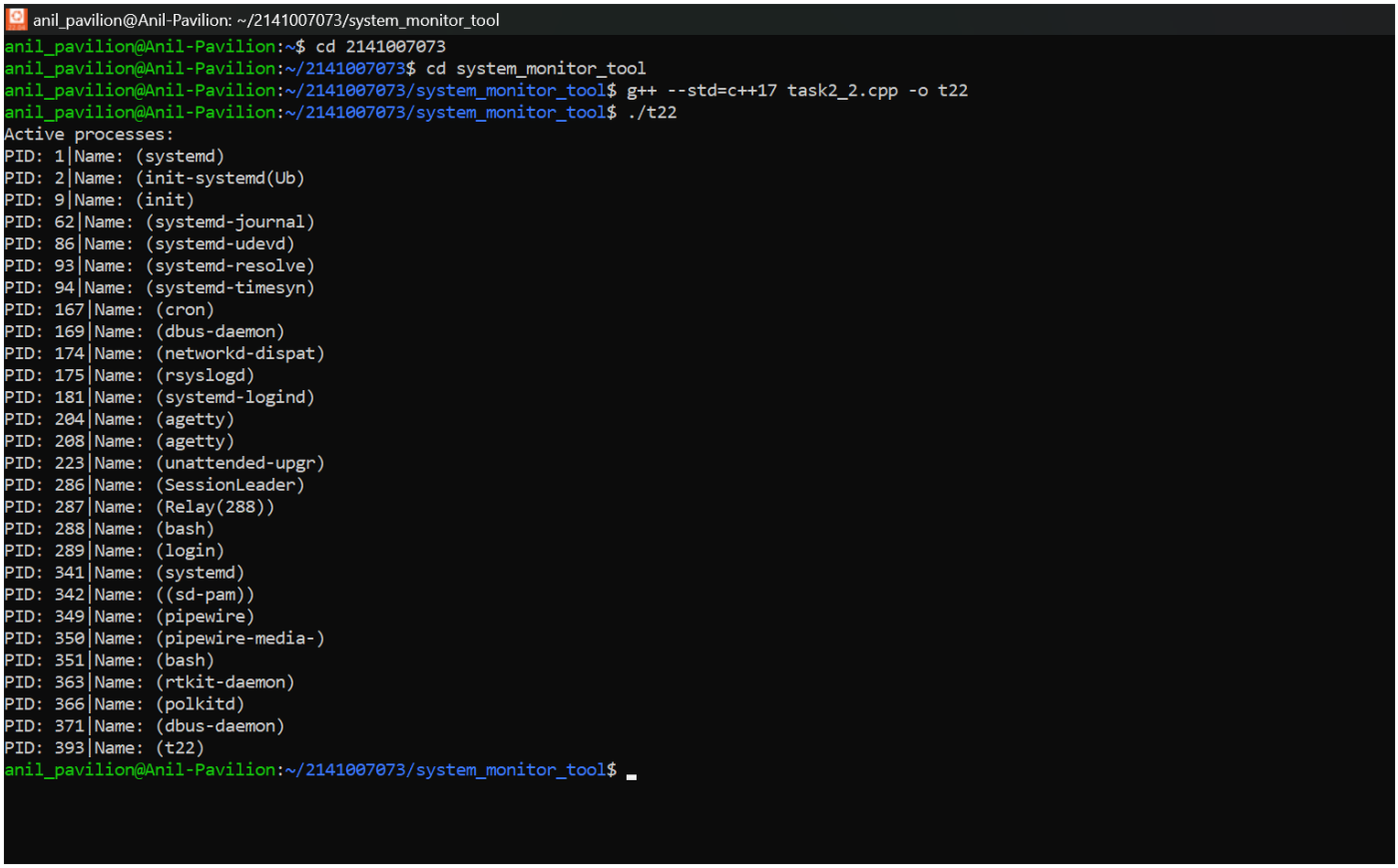
```
#include <iostream>
#include <fstream>
#include <sstream>
#include <thread>
#include <chrono>
using namespace std;
struct CPUData{
	long user, nice, system, idle, iowait, irq, softirq, steal, guest, guest_nice;
};
CPUData getCPUData(){
	ifstream file("/proc/stat");
	string line;
	CPUData cpu={};
	if (file.is_open()){
		getline(file,line);
		istringstream ss(line);
		string cpuLabel;
		ss >> cpuLabel >> cpu.user >> cpu.nice >> cpu.system >> cpu.idle >> cpu.iowait >> cpu.irq >>
cpu.softirq >> cpu.steal >> cpu.guest >> cpu.guest_nice;
	}
	return cpu;
}
double calculateCPUUsage (CPUData prev, CPUData current){
	long prevIdle= prev.idle + prev.iowait;
	long currIdle= current.idle + current.iowait;
	long prevTotal = prev.user + prev.nice + prev.system + prev.idle + prev.iowait + prev.irq + prev.softirq +
prev.steal;
	long currTotal = current.user + current.nice + current.system + current.idle + current.iowait + current.irq +
current.softirq + current.steal;
	long totalDiff = currTotal - prevTotal;
	long idleDiff = currIdle - prevIdle;
	return ((totalDiff - idleDiff) * 100.0) / totalDiff;
}
int main(){
	CPUData cpu= getCPUData();
	cout << "User: " << cpu.user << endl;
   cout << "Nice: " << cpu.nice << endl;
   cout << "System: " << cpu.system << endl;
   cout << "Idle: " << cpu.idle << endl;
   cout << "IOwait: " << cpu.iowait << endl;
   cout << "IRQ: " << cpu.irq << endl;
   cout << "SoftIRQ: " << cpu.softirq << endl;
   cout << "Steal: " << cpu.steal << endl;
   cout << "Guest: " << cpu.guest << endl;
   cout << "Guest Nice: " << cpu.guest_nice << endl;
   CPUData prevData = getCPUData();
   this_thread::sleep_for(chrono::seconds(1));
   CPUData currData = getCPUData();
   double cpuUsage= calculateCPUUsage (prevData, currData);
   cout << "CPU Usage: "<<cpuUsage<<endl;
	return 0;
}
```

```
anil_pavilion@Anil-Pavilion: ~/2141007073/system_monitor_tool
anil_pavilion@Anil-Pavilion:~/2141007073/system_monitor_tool$ g++ --std=c++17 task2.cpp -o t2
anil_pavilion@Anil-Pavilion:~/2141007073/system_monitor_tool$ ./t2
User: 610
Nice: 0
System: 714
Idle: 737210
IOwait: 471
IRQ: 0
SoftIRQ: 207
Steal: 0
Guest: 0
Guest Nice: 0
CPU Usage: 0.0832639
anil_pavilion@Anil-Pavilion:~/2141007073/system_monitor_tool$
```

# Part 2

```
#include <iostream>
#include <filesystem>
#include <fstream>
#include <sstream>
#include <algorithm>
using namespace std;
namespace fs = std::filesystem;
bool isNumber(const string &s){
	return  !s.empty() && all_of(s.begin(), s.end(), ::isdigit);
}

string getProcessName(int pid){
	ifstream file("/proc/" + to_string(pid) + "/stat");
	string line, processName;
	if(file.is_open()){
		getline(file,line);
		istringstream ss(line);
		string token;
		int count=0;
		while(ss >> token){
			count++;
			if(count==2){
				processName=token;
				break;
			}
		}
	}
	return processName;
}
```

```
int main(){
	cout<<"Active processes: "<<endl;
	for (const auto &entry : fs::directory_iterator("/proc")){
		if(entry.is_directory()){
			string filename=entry.path().filename().string();
			if (isNumber(filename)){
				int pid=stoi(filename);
				string processName = getProcessName(pid);
				cout<<"PID: "<<pid<<"|Name: " << processName<<endl;
			}
		}
	}
	return 0;
}
```




---

## Task - 3

```
#include <iostream>
#include <filesystem>
#include <fstream>
#include <sstream>
#include <algorithm>
```

```
#include <vector>
#include <dirent.h>
#include <unistd.h>
#include <csignal>
using namespace std;
namespace fs=std::filesystem;
struct ProcessInfo{
	int pid;
	string name;
	double cpuUsage;
	long memoryUsage;
};
string readFileValue(const string &path){
	ifstream file(path);
	string value;
	if (file.is_open()){
		getline(file,value);
	}
	return value;
}
double getSystemUptime(){
	ifstream file("/proc/uptime");
	double uptime;
	if (file.is_open()){
		file >> uptime;
	}
	return uptime;
}
ProcessInfo getProcessInfo(int pid,double systemUptime){
	ProcessInfo pinfo;
   pinfo.pid = pid;
	ifstream file("/proc/" + to_string(pid) + "/stat");
	string line;
	long utime=0, stime=0, starttime=0;
	if(file.is_open()){
		getline(file,line);
		istringstream ss(line);
		string token;
		int count=0;

		while(ss >> token){
			count++;
			if(count==2) pinfo.name=token;
			else if(count==14) utime=stol(token);
			else if(count==15) stime=stol(token);
			else if(count==22) starttime=stol(token);
		}
	}

	ifstream memFile("/proc/" + to_string(pid) + "/status");
	if (memFile.is_open()){
		string key, value,unit;
		while(memFile >> key >> value >> unit){
			if (key=="VmRSS"){
				pinfo.memoryUsage=stol(value);
				break;
			}
```

```
		}
	}
	long total_time = utime + stime;
	double seconds = systemUptime - (starttime / sysconf(_SC_CLK_TCK));
	pinfo.cpuUsage = (total_time / sysconf(_SC_CLK_TCK)) / seconds * 100;
	return pinfo;


}

vector <ProcessInfo> getAllProcesses(){
	vector <ProcessInfo> processes;
	double systemUptime= getSystemUptime();
	for (const auto &entry : fs::directory_iterator("/proc")){
		if (entry.is_directory()){
			string filename=entry.path().filename().string();
			if (all_of(filename.begin(),filename.end(), ::isdigit)){
				int pid=stoi(filename);
				processes.push_back(getProcessInfo(pid,systemUptime));
			}
		}
	}
	return processes;
}

void sortProcesses(vector<ProcessInfo> &processes, bool sortByCpu){
	if(sortByCpu){
		sort(processes.begin(),processes.end(),[](const ProcessInfo &a, const ProcessInfo &b)
		{
			return a.cpuUsage > b.cpuUsage;
		});
	}
	else{
		sort(processes.begin(),processes.end(),[](const ProcessInfo &a, const ProcessInfo &b)
		{
			return a.memoryUsage > b.memoryUsage;
		});
	}
}


int main(){
	vector <ProcessInfo> processes = getAllProcesses();
	sortProcesses(processes,true);
	cout<<"PID\tCPU%\tMemory (kB)\tName\n";
	for (size_t i=0;i<min(processes.size(),size_t(10));i++){

	cout<<processes[i].pid<<"\t"<<processes[i].cpuUsage<<"%\t"<<processes[i].memoryUsage<<"%\t"<<process
es[i].name<<"%\n";
	}
	return 0;
}
```

```
anil_pavilion@Anil-Pavilion: ~/2141007073/system_monitor_tool
anil_pavilion@Anil-Pavilion:~/2141007073/system_monitor_tool$ g++ --std=c++17 task3.cpp -o t3
anil_pavilion@Anil-Pavilion:~/2141007073/system_monitor_tool$ ./t3
PID     CPU%    Memory (kB)     Name
223     0%      140363473426848%        (unattended-upgr)%
407     0%      140363473426848%        (t3)%
371     0%      140363473426848%        (dbus-daemon)%
366     0%      140363473426848%        (polkitd)%
363     0%      140363473426848%        (rtkit-daemon)%
351     0%      140363473426848%        (bash)%
350     0%      140363473426848%        (pipewire-media-)%
349     0%      140363473426848%        (pipewire)%
342     0%      140363473426848%        ((sd-pam))%
341     0%      140363473426848%        (systemd)%
anil_pavilion@Anil-Pavilion:~/2141007073/system_monitor_tool$
```

---

## Task - 4

```
#include <iostream>
#include <filesystem>
#include <fstream>
#include <sstream>
#include <algorithm>
#include <vector>
#include <dirent.h>
#include <unistd.h>
#include <csignal>
using namespace std;
namespace fs=std::filesystem;
struct ProcessInfo{
        int pid;
        string name;
        double cpuUsage;
        long memoryUsage;
};
string readFileValue(const string &path){
```

```
	ifstream file(path);
	string value;
	if (file.is_open()){
		getline(file,value);
	}
	return value;
}
double getSystemUptime(){
	ifstream file("/proc/uptime");
	double uptime;
	if (file.is_open()){
		file >> uptime;
	}
	return uptime;
}
ProcessInfo getProcessInfo(int pid,double systemUptime){
	ProcessInfo pinfo;
   pinfo.pid = pid;
	ifstream file("/proc/" + to_string(pid) + "/stat");
	string line;
	long utime=0, stime=0, starttime=0;
	if(file.is_open()){
		getline(file,line);
		istringstream ss(line);
		string token;
		int count=0;

		while(ss >> token){
			count++;
			if(count==2) pinfo.name=token;
			else if(count==14) utime=stol(token);
			else if(count==15) stime=stol(token);
			else if(count==22) starttime=stol(token);
		}
	}

	ifstream memFile("/proc/" + to_string(pid) + "/status");
	if (memFile.is_open()){
		string key, value,unit;
		while(memFile >> key >> value >> unit){
			if (key=="VmRSS"){
				pinfo.memoryUsage=stol(value);
				break;
			}
		}
	}
	long total_time = utime + stime;
	double seconds = systemUptime - (starttime / sysconf(_SC_CLK_TCK));
	pinfo.cpuUsage = (total_time / sysconf(_SC_CLK_TCK)) / seconds * 100;
	return pinfo;


}

vector <ProcessInfo> getAllProcesses(){
	vector <ProcessInfo> processes;
	double systemUptime= getSystemUptime();
```

```
	for (const auto &entry : fs::directory_iterator("/proc")){
		if (entry.is_directory()){
			string filename=entry.path().filename().string();
			if (all_of(filename.begin(),filename.end(), ::isdigit)){
				int pid=stoi(filename);
				processes.push_back(getProcessInfo(pid,systemUptime));
			}
		}
	}
	return processes;
}

void sortProcesses(vector<ProcessInfo> &processes, bool sortByCpu){
	if(sortByCpu){
		sort(processes.begin(),processes.end(),[](const ProcessInfo &a, const ProcessInfo &b)
		{
			return a.cpuUsage > b.cpuUsage;
		});
	}
	else{
		sort(processes.begin(),processes.end(),[](const ProcessInfo &a, const ProcessInfo &b)
		{
			return a.memoryUsage > b.memoryUsage;
		});
	}
}


int main(){
	vector <ProcessInfo> processes = getAllProcesses();
	sortProcesses(processes,true);
	cout<<"PID\tCPU%\tMemory (kB)\tName\n";
	for (size_t i=0;i<min(processes.size(),size_t(10));i++){

	cout<<processes[i].pid<<"\t"<<processes[i].cpuUsage<<"%\t"<<processes[i].memoryUsage<<"%\t"<<process
es[i].name<<"%\n";
	}
	int targetPid;
	cout<<"Enter PID to kill: ";
	cin>>targetPid;
	if(targetPid > 0){
		if (kill(targetPid, SIGKILL) == 0){
			cout<<"Process "<< targetPid << " terminated successfully\n";
		}
		else{
			perror("Failed to kill the process");
		}
	}
	return 0;
}
```

```
anil_pavilion@Anil-Pavilion: ~/2141007073/system_monitor_tool
anil_pavilion@Anil-Pavilion:~/2141007073/system_monitor_tool$ g++ --std=c++17 task4.cpp -o t4
anil_pavilion@Anil-Pavilion:~/2141007073/system_monitor_tool$ ./t4
PID     CPU%    Memory (kB)     Name
223     0%      0%      (unattended-upgr)%
428     0%      0%      (t4)%
371     0%      0%      (dbus-daemon)%
366     0%      0%      (polkitd)%
363     0%      0%      (rtkit-daemon)%
351     0%      0%      (bash)%
350     0%      0%      (pipewire-media-)%
349     0%      0%      (pipewire)%
342     0%      0%      ((sd-pam))%
341     0%      0%      (systemd)%
Enter PID to kill: 350
Process 350 terminated successfully
anil_pavilion@Anil-Pavilion:~/2141007073/system_monitor_tool$
```

---

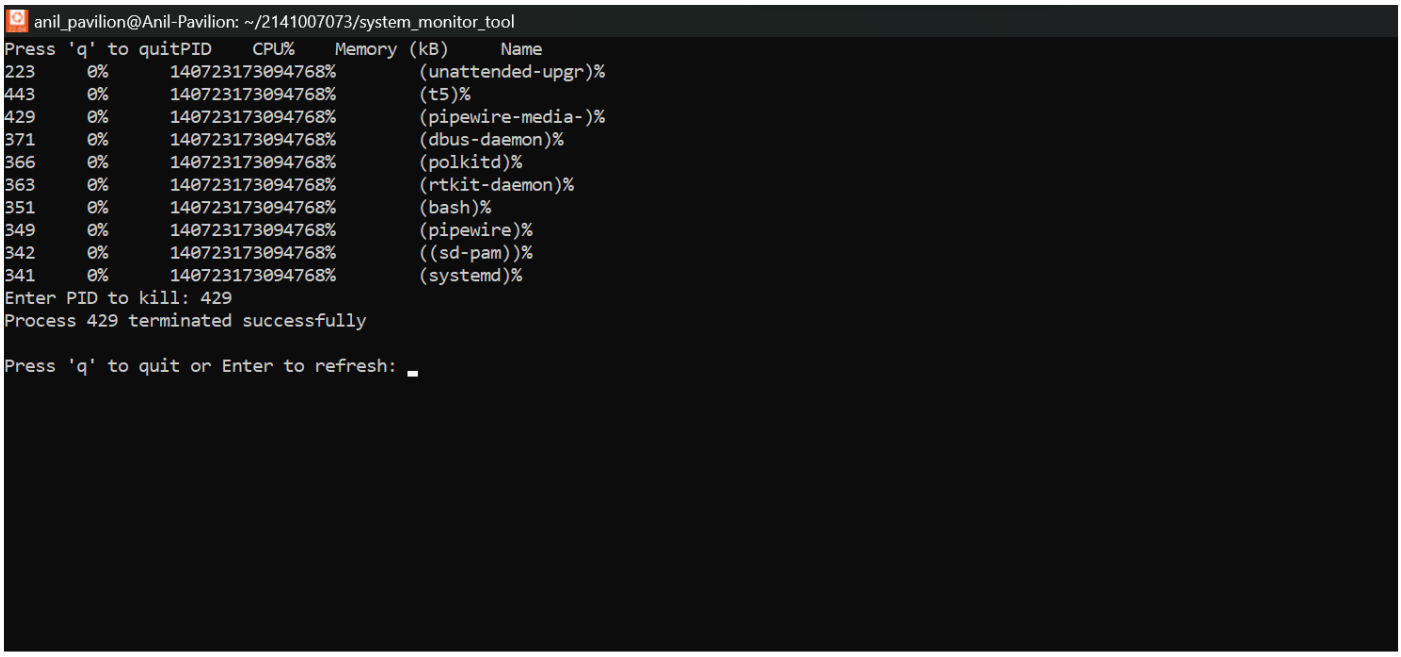
# Task - 5

```
#include <iostream>
#include <filesystem>
#include <fstream>
#include <sstream>
#include <algorithm>
#include <vector>
#include <dirent.h>
#include <unistd.h>
#include <csignal>
#include <thread>
#include <chrono>
using namespace std;
namespace fs=std::filesystem;
struct ProcessInfo{
        int pid;
        string name;
        double cpuUsage;
        long memoryUsage;
};
string readFileValue(const string &path){
        ifstream file(path);
        string value;
```

```
	if (file.is_open()){
		getline(file,value);
	}
	return value;
}
double getSystemUptime(){
	ifstream file("/proc/uptime");
	double uptime;
	if (file.is_open()){
		file >> uptime;
	}
	return uptime;
}
ProcessInfo getProcessInfo(int pid,double systemUptime){
	ProcessInfo pinfo;
   pinfo.pid = pid;
	ifstream file("/proc/" + to_string(pid) + "/stat");
	string line;
	long utime=0, stime=0, starttime=0;
	if(file.is_open()){
		getline(file,line);
		istringstream ss(line);
		string token;
		int count=0;

		while(ss >> token){
			count++;
			if(count==2) pinfo.name=token;
			else if(count==14) utime=stol(token);
			else if(count==15) stime=stol(token);
			else if(count==22) starttime=stol(token);
		}
	}

	ifstream memFile("/proc/" + to_string(pid) + "/status");
	if (memFile.is_open()){
		string key, value,unit;
		while(memFile >> key >> value >> unit){
			if (key=="VmRSS"){
				pinfo.memoryUsage=stol(value);
				break;
			}
		}
	}
	long total_time = utime + stime;
	double seconds = systemUptime - (starttime / sysconf(_SC_CLK_TCK));
	pinfo.cpuUsage = (total_time / sysconf(_SC_CLK_TCK)) / seconds * 100;
	return pinfo;


}

vector <ProcessInfo> getAllProcesses(){
	vector <ProcessInfo> processes;
	double systemUptime= getSystemUptime();
	for (const auto &entry : fs::directory_iterator("/proc")){
		if (entry.is_directory()){
```

```
                string filename=entry.path().filename().string();
                if (all_of(filename.begin(),filename.end(), ::isdigit)){
                        int pid=stoi(filename);
                        processes.push_back(getProcessInfo(pid,systemUptime));
                }
        }
    }
    return processes;
}

void sortProcesses(vector<ProcessInfo> &processes, bool sortByCpu){
    if(sortByCpu){
        sort(processes.begin(),processes.end(),[](const ProcessInfo &a, const ProcessInfo &b)
        {
                return a.cpuUsage > b.cpuUsage;
        });
    }
    else{
        sort(processes.begin(),processes.end(),[](const ProcessInfo &a, const ProcessInfo &b)
        {
                return a.memoryUsage > b.memoryUsage;
        });
    }
}


int main(){
    char input;
    while (true){
        system("clear");
        vector <ProcessInfo> processes = getAllProcesses();
        sortProcesses(processes,true);
        cout<<"Press 'q' to quit";
        cout<<"PID\tCPU%\tMemory (kB)\tName\n";
        for (size_t i=0;i<min(processes.size(),size_t(10));i++){

    cout<<processes[i].pid<<"\t"<<processes[i].cpuUsage<<"%\t"<<processes[i].memoryUsage<<"%\t"<<process
es[i].name<<"%\n";
        }
        int targetPid;
        cout<<"Enter PID to kill: ";
        cin>>targetPid;
        if (targetPid == 0){
                //continue refresh
        }
        else if(targetPid > 0){
                if (kill(targetPid, SIGKILL) == 0){
                        cout<<"Process "<< targetPid << " terminated successfully\n";
                }
                else{
                        perror("Failed to kill the process");
                }
        }

        cout << "\nPress 'q' to quit or Enter to refresh: ";
        cin.ignore( );
        input = getchar( );
```

```
            if (input == 'q' || input =='Q')
            break;
            this_thread::sleep_for(chrono::seconds(2));
        }
        return 0;
}
```




---

**Name:** Anil Gupta
**Reg. No.:** 2141007073