Q1:

Copying the header and .cpp code snippets of moment-gatherer class:

```
//5.1
class StatisticsMoments4 : public StatisticsMC
{

public:

    StatisticsMoments4();
    virtual void DumpOneResult(double result);
    virtual std::vector<std::vector<double> > GetResultsSoFar() const;
    virtual StatisticsMC* clone() const;

private:

    double RunningSum;
    double RunningSum2;
    double RunningSum3;
    double RunningSum4;
    unsigned long PathsDone;

};
```

Header file

```
StatisticsMC* StatisticsMean::clone()const
{
    return new StatisticsMean(*this);
}

StatisticsMoments4::StatisticsMoments4()
:RunningSum(0.0), RunningSum2(0.0),RunningSum3(0.0),RunningSum4(0.0),PathsDone(0UL)
{}

void StatisticsMoments4::DumpOneResult(double result)
{
    PathsDone++;
    RunningSum +=  result;
    RunningSum2 +=  result* result;
    RunningSum3 += result * result * result;
    RunningSum4 += result * result * result * result;
}
// Calculating moments here
vector<vector<double> > StatisticsMoments4::GetResultsSoFar() const
{
    vector<vector<double> > Results(1);

    Results[0].resize(4);
    Results[0][0] = RunningSum / PathsDone;
    Results[0][1] = RunningSum2 / PathsDone;
    Results[0][2] = RunningSum3 / PathsDone;
    Results[0][3] = RunningSum4 / PathsDone;

    return Results;
}

StatisticsMC* StatisticsMoments4::clone() const
{
    return new StatisticsMoments4(*this);
}
```

.cpp definition of functions.

Q2:

Copying the header and .cpp code snippets of VAR-gatherer class:

Note on VAR: If we have generated N scenarios, and want to calculate VAR at an alpha confidence level, we can sort the m scenarios, V1,V2,V3,….Vm.

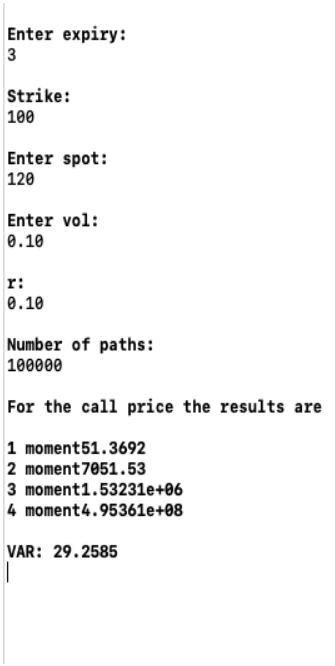
Then VaR = Vk, where k= alpha*m.

```
//5.2
class ValueAtRisk : public StatisticsMC
{

public:

    ValueAtRisk(double alpha_);
    virtual void DumpOneResult(double result);
    virtual std::vector<std::vector<double> > GetResultsSoFar() const;
    virtual StatisticsMC* clone() const;

private:

    std::vector<double> PathData;
    double alpha;
    unsigned long PathsDone;

};                    //Header



ValueAtRisk::ValueAtRisk(double alpha_) : alpha(alpha_)
{
    PathsDone=0;
}

StatisticsMC* ValueAtRisk::clone() const
{
    return new ValueAtRisk(*this);
}

void ValueAtRisk::DumpOneResult(double result)
{
    PathData.push_back(result);
    ++PathsDone;
}

vector<vector<double> >  ValueAtRisk::GetResultsSoFar() const
{
    vector<vector<double> > Results(1);
    Results[0].resize(1);

    vector<double> tmp(PathData);

    sort(tmp.begin(), tmp.end());
    int n= int(tmp.size());
    int var_slot((int)(ceil(n*alpha)));

    Results[0][0] = tmp[var_slot];
    return Results;
}


         //.cpp definition
```

Main.cpp

```
    PayOffCall thePayOff(Strike);
    VanillaOption theOption(thePayOff, Expiry);
    ParametersConstant VolParam(Vol);
    ParametersConstant rParam(r);
    StatisticsMoments4 momentgatherer;     // Moments gatherer object
    ValueAtRisk        riskgatherer(0.5); // aplha shows : confidence level

    SimpleMonteCarlo5(theOption, Spot, VolParam, rParam, NumberOfPaths, momentgatherer);// calling with moment-gatherer
    SimpleMonteCarlo5(theOption, Spot, VolParam, rParam, NumberOfPaths, riskgatherer); //  calling with risk-gatherer

    vector<vector<double> > results_moment = momentgatherer.GetResultsSoFar();

    vector<vector<double> > results_VAR = riskgatherer.GetResultsSoFar();

    cout <<"\nFor the call price the results are \n\n";

    // printing result of Moments
    for (unsigned long i=0; i < results_moment.size(); i++)
    {
        for (unsigned long j=0; j < results_moment[i].size(); j++)
            cout <<j+1<<" moment" <<results_moment[i][j] << "\n";
        cout << "\n";
    }
    // printing result of VAR
    for (unsigned long i=0; i < results_VAR.size(); i++)
    {
        for (unsigned long j=0; j < results_VAR[i].size(); j++)
            cout <<"VAR: " <<results_VAR[i][j] << " ";
        cout << "\n";
    }
```

Output:



```
Enter expiry:
3

Strike:
100

Enter spot:
120

Enter vol:
0.10

r:
0.10

Number of paths:
100000

For the call price the results are

1 moment51.3692
2 moment7051.53
3 moment1.53231e+06
4 moment4.95361e+08

VAR: 29.2585
```