Question 1 :

Antithetic Sampling : We can purposely try to induce negative correlation among the variables X1, . . . , Xn, or generate copies that while having the same mean, have a smaller variance, so that the variance of the estimator in X1 = sum(Xi,n), becomes smaller than (σ^2)/ n , resulting in a smaller confidence interval.The idea is to try to get even better estimates by reducing the uncertainty in our estimate. Antithetic sampling works like this :

Suppose you would like to estimate, X = E(Y), therefore for Y1,Y2 .

The unbiased estimate will be X = [E(Y1) + E(Y2)]/2.

And the variance will be var = [Var(Y1)+Var(Y2) + 2*Covar(Y1,Y2)]/2

So the variance is reduced when the Covariance is reduced, that's why we take samples like this : {Y1,Y2,.....} and {-Y1, -Y2}.

Experiment results : With Antithetic the value started converging from 256 paths while without antithetic results it took 2048 paths

CALL

Enter expiry:2 **Enter Strike:**100 **Enter spot:** 120 **Enter vol:** 0.13 **Enter r:** 0.10 **Enter Number of paths:** 100000

Antithetic- Num of Paths	**Without Antithetic - Num of Paths**
89.5849 2 **64.1036 4** **50.3544 8** **44.1451 16** **41.3126 32** **39.8516 64** **38.9911 128** **38.9035 256** **38.6539 512** **38.3874 1024** **38.2028 2048** **38.279 4096** **38.2924 8192** **38.3219 16384** **38.2954 32768** **38.248 65536** **38.248 100000**	**7.08699 2** **25.0471 4** **26.2534 8** **31.4175 16** **35.4511 32** **37.8828 64** **38.5391 128** **39.3641 256** **37.7306 512** **37.8539 1024** **38.3688 2048** **38.1529 4096** **38.4519 8192** **38.37 16384** **38.2571 32768** **38.2953 65536** **38.2603 100000**

Q2 :

The Boost Library use the Mersenne Twister as a random generator, which is highly regarded for its performance and high quality pseudo-random numbers. It was designed with statistical simulations in mind, and should therefore be quite good for Monte Carlo simulations, probabilistic algorithms and so on.

Creating the MT random generator on the lines of park miller , i.e inheriting RandomBase class, and using the adapter pattern.To use the adapter pattern simply means to use an intermediary class which transforms one interface into another.

```
42
43  class RandomMersenneTwister : public RandomBase
44  {
45  public:
46
47      RandomMersenneTwister(unsigned long Dimensionality, unsigned long Seed = 12411);
48
49      virtual RandomBase* clone() const;
50
51      virtual void Skip(unsigned long numberOfPaths);
52      virtual void SetSeed(unsigned long Seed);
53      virtual void Reset();
54      virtual void ResetDimensionality(unsigned long NewDimensionality);
55
56      virtual void GetUniforms(MJArray& variates);
57      virtual void GetUniforms(vector<double>& variates, double a = 0.0, double b = 1.0);
58
59      virtual void GetGaussians(MJArray& variates);
60      virtual void GetGaussians(vector<double>& variates, double m = 0.0, double s = 1.0);
61
62      virtual void GetLogNormals(double m, double s, vector<double>& variates);
63      virtual void GetExponentials(double lambda, vector<double>& variates);
64      virtual void GetPoissons(double lambda, vector<double>& variates);
65      virtual void GetGammas(double alpha, double beta, vector<double>& variates);
66      virtual void GetBetas(double alpha, double beta, vector<double>& variates);
67
68  private:
69      MersenneTwister InnerGenerator;
70      unsigned long InitialSeed;
71  };
72  #endif /* MersenneTwister_h */
73
```

Verified by validating the the Call option values generated from ParkMuller and Mersene Twister.

```
79
80
81
82      RandomParkMiller generator(1);
83
84      RandomMersenneTwister  generator1(1);
85
86      AntiThetic GenTwo(generator);
87
88      // for call option
89      SimpleMonteCarlo6(theOption,
90                        Spot,
91                        VolParam,
92                        rParam,
93                        NumberOfPaths,
94                        gathererTwo,
95                        generator);
96      vector<vector<double> > results = gathererTwo.GetResultsSoFar();
97
98      cout <<"\nFor the call price the results are \n";
99      for (unsigned long i=0; i < results.size(); i++)
100     {
101         for (unsigned long j=0; j < results[i].size(); j++)
102             cout << results[i][j] << " ";
103         cout << "\n";
104     }
105
106
107
108     double tmp;
109     cin >> tmp;
110
111     return 0;
112
113 }
114
```

Values :

MT	ParkMuller
For the call price the results are 35.8481 2 35.3401 4 36.1903 8 40.3633 16 39.0092 32 45.3174 64 45.6874 128 44.7761 256 44.5549 512 42.7831 1024 42.7362 2048 43.1412 4096 43.0818 8192	For the call price the results are 10.3592 2 28.9389 4 30.999 8 36.2255 16 40.0105 32 42.464 64 43.0846 128 43.8838 256 42.3815 512 42.5391 1024 43.0221 2048 42.8154 4096 43.0918 8192

42.9845 16384 **43.0679 32768** **42.9823 65536** **42.9914 100000**	**43.0219 16384** **42.9254 32768** **42.9556 65536** **42.9258 100000**