Listing 2.1 BinModel02.h

```
#ifndef BinModel02_h
#define BinModel02_h
class BinModel
{
private:
double S0;
double U;
double D;
double R;
public:
double RiskNeutProb();
double S(int n, int i);
int GetInputData();
double GetR();
};
#endif
```

Listing 2.2 BinModel02.cpp

```
#include "BinModel02.h"
#include <iostream>
#include <cmath>
using namespace std;
double BinModel::RiskNeutProb()
{
return (R-D)/(U-D);
}
double BinModel::S(int n, int i)
{
return S0*pow(1+U,i)*pow(1+D,n-i);
}
int BinModel::GetInputData()
{
cout << "Enter S0: "; cin >> S0;
cout << "Enter U: "; cin >> U;
cout << "Enter D: "; cin >> D;
cout << "Enter R: "; cin >> R;
cout << endl;
if (S0<=0.0 || U<=-1.0 || D<=-1.0 || U<=D
|| R<=-1.0)
{
cout << "Illegal data ranges" << endl;
cout << "Terminating program" << endl;
return 1;
}
if (R>=U || R<=D)
{
cout << "Arbitrage exists" << endl;
cout << "Terminating program" << endl;
return 1;
}
cout << "Input data checked" << endl;
cout << "There is no arbitrage" << endl << endl;
return 0;
```

```
}
double BinModel::GetR()
{
return R;
}
```

Listing 2.3 Options04.h

```
#ifndef Options04_h
#define Options04_h
#include "BinModel02.h"
int GetInputData(int& N, double& K);
double PriceByCRR(BinModel Model, int N, double K,
double (*Payoff)(double z, double K));
double CallPayoff(double z, double K);
double PutPayoff(double z, double K);
#endif
```

Listing 2.4 Options04.cpp

```
#include "Options04.h"
#include "BinModel02.h"
#include <iostream>
#include <cmath>
using namespace std;
int GetInputData(int& N, double& K)
{
cout << "Enter steps to expiry N: "; cin >> N;
cout << "Enter strike price K: "; cin >> K;
cout << endl;
return 0;
}
double PriceByCRR(BinModel Model, int N, double K,
double (*Payoff)(double z, double K))
{
double q=Model.RiskNeutProb();
double Price[N+1];
for (int i=0; i<=N; i++)
{
Price[i]=Payoff(Model.S(N,i),K);
}
for (int n=N-1; n>=0; n--)
{
for (int i=0; i<=n; i++)
{
Price[i]=(q*Price[i+1]+(1-q)*Price[i])
/(1+Model.GetR());
}
}
return Price[0];
}
double CallPayoff(double z, double K)
{
if (z>K) return z-K;
return 0.0;
}
```

```
double PutPayoff(double z, double K)
{
if (z<K) return K-z;
return 0.0;
}
```

Listing 2.5 Main08.cpp

```
#include "BinModel02.h"
#include "Options04.h"
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
BinModel Model;
if (Model.GetInputData()==1) return 1;
double K;
int N;
cout << "Enter call option data:" << endl;
GetInputData(N,K);
cout << "European call option price = "
<< PriceByCRR(Model,N,K,CallPayoff)
<< endl << endl;
cout << "Enter put option data:" << endl;
GetInputData(N,K);
cout << "European put option price = "
<< PriceByCRR(Model,N,K,PutPayoff)
<< endl << endl;
return 0;
}
```

Listing 2.6 Options05.h

```
#ifndef Options05_h
#define Options05_h
#include "BinModel02.h"
class EurOption
{
private:
int N;
double (*Payoff)(double z, double K);
public:
void SetN(int N_){N=N_;}
void SetPayoff
(double (*Payoff_)(double z, double K))
{Payoff=Payoff_;}
double PriceByCRR(BinModel Model, double K);
};
double CallPayoff(double z, double K);
class Call: public EurOption
{
private:
double K;
public:
```

```
Call(){SetPayoff(CallPayoff);}
double GetK(){return K;}
int GetInputData();
};
double PutPayoff(double z, double K);
class Put: public EurOption
{
private:
double K;
public:
Put(){SetPayoff(PutPayoff);}
double GetK(){return K;}
int GetInputData();
};
#endif
```

Listing 2.7 Options05.cpp

```
#include "Options05.h"
#include "BinModel02.h"
#include <iostream>
#include <cmath>
using namespace std;
double EurOption::PriceByCRR(BinModel Model, double K)
{
double q=Model.RiskNeutProb();
double Price[N+1];
for (int i=0; i<=N; i++)
{
Price[i]=Payoff(Model.S(N,i),K);
}
for (int n=N-1; n>=0; n--)
{
for (int i=0; i<=n; i++)
{
Price[i]=(q*Price[i+1]+(1-q)*Price[i])
/(1+Model.GetR());
}
}
return Price[0];
}
double CallPayoff(double z, double K)
{
if (z>K) return z-K;
return 0.0;
}
int Call::GetInputData()
{
cout << "Enter call option data:" << endl;
int N;
cout << "Enter steps to expiry N: "; cin >> N;
SetN(N);
cout << "Enter strike price K: "; cin >> K;
cout << endl;
return 0;
```

```
}
double PutPayoff(double z, double K)
{
if (z<K) return K-z;
return 0.0;
}
int Put::GetInputData()
{
cout << "Enter put option data:" << endl;
int N;
cout << "Enter steps to expiry N: "; cin >> N;
SetN(N);
cout << "Enter strike price K: "; cin >> K;
cout << endl;
return 0;
}
```

Listing 2.8 Main09.cpp

```
#include "BinModel02.h"
#include "Options05.h"
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
BinModel Model;
if (Model.GetInputData()==1) return 1;
Call Option1;
Option1.GetInputData();
cout << "European call option price = "
<< Option1.PriceByCRR(Model,Option1.GetK())
<< endl << endl;
Put Option2;
Option2.GetInputData();
cout << "European put option price = "
<< Option2.PriceByCRR(Model,Option2.GetK())
<< endl << endl;
return 0;
}
```

Listing 2.9 Options06.h

```
#ifndef Options06_h
#define Options06_h
#include "BinModel02.h"
class EurOption
{
private:
int N;
public:
void SetN(int N_){N=N_;}
virtual double Payoff(double z){return 0.0;}
```

```
double PriceByCRR(BinModel Model);
};
class Call: public EurOption
{
private:
double K;
public:
void SetK(double K_){K=K_;}
int GetInputData();
double Payoff(double z);
};
class Put: public EurOption
{
private:
double K;
public:
void SetK(double K_){K=K_;}
int GetInputData();
double Payoff(double z);
};
#endif
```

Listing 2.10 Options06.cpp

```
#include "Options06.h"
#include "BinModel02.h"
#include <iostream>
#include <cmath>
using namespace std;
double EurOption::PriceByCRR(BinModel Model)
{
double q=Model.RiskNeutProb();
double Price[N+1];
for (int i=0; i<=N; i++)
{
Price[i]=Payoff(Model.S(N,i));
}
for (int n=N-1; n>=0; n--)
{
for (int i=0; i<=n; i++)
{
Price[i]=(q*Price[i+1]+(1-q)*Price[i])
/(1+Model.GetR());
}
}
return Price[0];
}
int Call::GetInputData()
{
cout << "Enter call option data:" << endl;
int N;
cout << "Enter steps to expiry N: "; cin >> N;
SetN(N);
cout << "Enter strike price K: "; cin >> K;
cout << endl;
return 0;
```

```
}
double Call::Payoff(double z)
{
if (z>K) return z-K;
return 0.0;
}
int Put::GetInputData()
{
cout << "Enter put option data:" << endl;
int N;
cout << "Enter steps to expiry N: "; cin >> N;
SetN(N);
cout << "Enter strike price K: "; cin >> K;
cout << endl;
return 0;
}
double Put::Payoff(double z)
{
if (z<K) return K-z;
return 0.0;
}
```

Listing 2.11 Main10.cpp

```
#include "BinModel02.h"
#include "Options06.h"
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
BinModel Model;
if (Model.GetInputData()==1) return 1;
Call Option1;
Option1.GetInputData();
cout << "European call option price = "
<< Option1.PriceByCRR(Model)
<< endl << endl;
Put Option2;
Option2.GetInputData();
cout << "European put option price = "
<< Option2.PriceByCRR(Model)
<< endl << endl;
return 0;
}
```

Listing 2.12 DoubDigitOpt.h

```
#ifndef DoubDigitOpt_h
#define DoubDigitOpt_h
#include "Options06.h"
class DoubDigitOpt: public EurOption
{
private:
double K1;
double K2;
public:
```

```
int GetInputData();
double Payoff(double z);
};
#endif
```

Listing 2.13 DoubDigitOpt.cpp

```
#include "DoubDigitOpt.h"
#include <iostream>
using namespace std;
int DoubDigitOpt::GetInputData()
{
cout << "Enter double-digital option data:" << endl;
int N;
cout << "Enter steps to expiry N: "; cin >> N;
SetN(N);
cout << "Enter parameter K1: "; cin >> K1;
cout << "Enter parameter K2: "; cin >> K2;
cout << endl;
return 0;
}
double DoubDigitOpt::Payoff(double z)
{
if (K1<z && z<K2) return 1.0;
return 0.0;
}
```

Listing 2.14 Main11.cpp

```
#include "BinModel02.h"
#include "Options06.h"
#include "DoubDigitOpt.h"
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
BinModel Model;
if (Model.GetInputData()==1) return 1;
Call Option1;
Option1.GetInputData();
cout << "European call option price = "
<< Option1.PriceByCRR(Model)
<< endl << endl;
Put Option2;
Option2.GetInputData();
cout << "European put option price = "
<< Option2.PriceByCRR(Model)
<< endl << endl;
DoubDigitOpt Option3;
Option3.GetInputData();
cout << "European double-digital option price = "
<< Option3.PriceByCRR(Model)
<< endl << endl;
return 0;
}
```