```
//================================================================================
// Class to hold basic adult information
//================================================================================
class cs_adult {

    public:
    cs_adult(std::string, int, int);
    ~cs_adult();
    std::string getaname();
    int getati();
    void setati(int);
    int getaid();
    bool setrlt(int, int);
    void addrlt(int);
    bool rmvrlt(int);
    int rltsize();
    void setaname(std::string);
    std::string Display();

    private:
    std::string aname;
    int aid;
    int ati;
    std::deque<int> rlt;
};

//================================================================================
// Class to hold basic child information
//================================================================================
class cs_child {
    public:
    cs_child(std::string childs_name, int childs_age, int childs_cid);
    ~cs_child();
    std::string GetChildsName();
    int getcage();
    bool isteen();
    void setteen(bool);
    void addrlt(int);
    void addcare(int);
    bool setrlt(int, int);
    bool setcare(int, int);
    bool rmvrlt(int);
    bool rmvcare(int);
    int getrlt(int);
    int GetCare(int);
    int rltsize();
    void setcname(std::string);
    int parenttypecount(int);
    int caresz();
    bool care365();
    void setrltcounts(int, int, int, int, int);
    std::string Display();
    signed int GetTotalCare();

    private:
    std::string cname;
    int cid;
    int age;
    bool teen;
    RTYPES rltcounts;
    std::deque<int> rlt;
    std::deque<int> care;
};
//================================================================================
// Class for relationships
// a relationship will exist for every combination of child and adult.
// (i.e. number of relationships = number of children * number of adults.
//================================================================================
class cs_relationship {
    public:
        cs_relationship(int, int, int, int);
```

```
        cs_relationship(int, int, int , int , std::string);
        ~cs_relationship();
        int getchild();
        int GetAdult();
        DINT GetAdultandChild();
        int GetRelationshipType();
        int GetCare();
        int GetCarePercentage();
        int GetCostPercentage();
        std::string GetChildsName();
        bool IsInCase();
        void SetInCase(int);
        std::string Display();
    private:
        int child;
        int adult;
        int rtype;
        int ncare;
        bool incase;
        int owningcase;
        cs_carerates cr;
        std::string childname;
};
//=============================================================================
// Class for an adult case.
// An adult case reflects a sublevel case between an adult and a child, thus
// it mat be that there are multiple adult cases for the same adult that will
// be combined into a full case. There may also be additional adult cases as
// an adult can be involved in multiple full cases.
//=============================================================================
class cs_adultcase {
    public:
        cs_adultcase(int, int, std::deque<cs_adult>, std::deque<cs_relationship>);
        ~cs_adultcase();
        std::string GetAdultName();
        int GetRlt();           // the specific relationship object
        int GetRlTyp();         // the relationship type
        int GetAdult();
        int GetCare();
        int GetCarepc();
        int GetCostpc();
        int GetChild();
        std::string Display();
    private:
        int adult;
        int child;
        std::string aname;
        int rlt;
        int rltyp;
        int ati;
        int care;
        int carepc;
        int costpc;
        int mc_cost;
        int mc_cap;

};
//=============================================================================
// Class for each child case.
// A child case is a sublevel case which is used as the basis for a fullcase.
// A fullcase may have multiple children.
// A child case will only exist if the child has at least 1 parent and either
// another parent or a non-parent carer.
//=============================================================================
class cs_childcase {
    public:
        cs_childcase(int, std::deque<cs_child>);
        ~cs_childcase();
        int getchild();
        void AddAdultCase(int, std::deque<cs_adultcase>);
        void SetMainCarer(int, int);
```

```
        bool IsMerged();
        void SetMerged();
        std::string GetChildName();
        int GetMainCarerAdultCase();
        int GetMainCarerAdult();
        int GetMainCarerCare();
        int GetMainCarerType();
        int GetAdultCaseCount();
        int GetAdultCase(int);
        std::string Display();
        int GetACSize();
        int GetAC(int);
        bool IsThisChildName(std::string);
        void AddTotalRecipientCarePercent(int);
        int GetTotalRecipientCarePercent();
    private:
        std::string cname;                  // Name of the child
        int child;                      // index of the corresponding child
                                        // deque
        bool merged;                    // Flag to indicate whether or not
                                        // merged into a full case
        signed int main_carer_ac;       // index of the adult case deque entry
                                        // that has the most care of this child
        signed int main_carer_adult;    // index of the adult deque entry for
                                        // the adult with the most care
        signed int main_carer_typ;      // the relationship type of the main
                                        // carer
        signed int main_carer_care;     // the level of care of the main carer
        signed int total_recipient_carepc;  // The total percentage of care for
                                            //the CS recipients.
        std::deque<int> ac;
};
//=============================================================================
// Class for a full relationship
// A fullrelationship will exist for each fulladult/fullchildcase combination
// and is used to hold information at this sub level
//=============================================================================
class cs_full_adultcase;
class cs_fullcase;

class cs_full_relationship {
    public:
        cs_full_relationship(int, std::string, int, int, int, int, int, bool);
        std::string GetChildsName();
        int GetCare();
        int GetChild();
        int GetAdult();
        int GetRType();
        int GetCarePercentage();
        int GetCostPercentage();
        int GetCostOfTheChild();
        int GetMultiCaseCap();
        int GetMultiCaseCost();
        bool GetAge();
        void SetMCValues(int, int);
        void SetMCCap(int);
        int GetMCCost();
        void SetPCSPC(float);
        float GetPCSPC();
        float GetEffPCSPC();
        int GetCOTC();
        void SetCOTC(int);
        int GetPreAdjCOTC();
        void SetPreAdjCOTC(int);
        void AddRcvTotCare(int);
        int GetRcvTotCare();
        std::string Display();
        friend class cs_full_adultcase;
        friend class cs_fullcase;
    private:
        std::string childname;          // the child's name
```

```
        int child;                  // Reference to the child
        bool age;                   // Age Indicator (true = 13 or over)
        int adult;                  // reference to the adult
        int rtype;                  // Relationship type
        int care;                   // the level of care
                                    // (# of nights care the owning adult has)
        int carepc;                 // care percentage
        int costpc;                 // cost percentage (as per CS legilsation
                                    // e.g. 14%-under 35% = 24%)
        float pcspc;                // Parents child support percentage for
                                    // this child
        float effpcspc;             // Effective Parents child support
                                    // percentage (i.e if negatibe then 0)
        int preadj_cotc;            // Cost of the child before adjustments
        int cotc;                   // cost of this child
        int mccost;                 // MultiCase Cost of this child
        int mccap;                  // MultiCase Cap for this child;
        int rcv_tot_care;
};
//=============================================================================
// Class for a full adultcase
// A fulladult case represents the overall adult for a case
//=============================================================================
class cs_full_adultcase {
    public:
        cs_full_adultcase(int, std::string, int, int);
        ~cs_full_adultcase();
        int GetAdult();
        std::string GetAdultName();
        int GetATI();
        int GetATILessSSA();
        int GetSSA();
        int GetRDCCost();
        int GetATILessRDCCost();
        int GetRDCMinors();
        int GetRDCTeens();
        int GetMCMinors();
        int GetMCTeens();
        void AddFullRelationship(int, std::string, int, int, int, int, int,
                bool);
        void AddMCTeenRlt(int, std::string, int, int, int, int, int);
        void AddMCMinorRlt(int, std::string, int, int, int, int, int);
        int GetRltSize();
        int GetMCMinorsSize();
        int GetMCTeensSize();
        std::string GetRltChildsName(int);
        std::string GetMCTeensChildsName(int);
        std::string GetMCMinorsChildsName(int);
        int GetRltAdult(int);
        int GetRltChild(int);
        int GetRltCare(int);
        int GetRltRType(int);
        int GetRltCarePercentage(int);
        int GetRltCostPercentage(int);
        int GetRltCostOfTheChild(int);
        int GetRltMultiCaseCap(int);
        int GetRltMultiCaseCost(int);
        bool GetRltAge(int);
        void AddRelevantDependantMinor();
        void AddRelevantDependantTeen();
        void AddMultiCaseMinor();
        void AddMultiCaseTeen();
        void SetRDCCost(int);
        int GetMCMinorsCostPercentage(int);
        int GetMCTeensCostPercentage(int);
        void SetMCMinorsValues(int, int, int);
        void SetMCTeensValues(int, int, int);
        void SetMCCost(int);
        void SetMCCap(int);
        void SetRltMCCap(int, int);
        int GetMCMinorsMCCost(int);
```

```
        int GetMCTeensMCCost(int);
        std::string MCTeensDisplay(int);
        std::string MCMinorsDisplay(int);
        void SetPIP(float);
        int GetMCA();
        int GetMCCap();
        int GetATILessMCA();
        int GetCSI();
        float GetPIP();
        void SetRltPCSPC(int, float);
        float GetRltPCSPC(int);
        float GetRltEffPCSPC(int);
        int GetRltCOTC(int);
        void SetRltCOTC(int, int);
        int GetRltPreAdjCOTC(int);
        void SetRltPreAdjCOTC(int, int);
        void AddCSCost(float);
        void AddCSGets(float);
        float GetCSGets();
        float GetPays();
        void AddRltRcvTotCare(int, int);
        int GetRltRcvTotCare(int);
        friend class cs_fullcase;
    private:
        int fullcase;           // Identifier (unused at present)
        int adult;              // The adult
        std::string aname;           // The adult's name
        int rdc_minors;         // Thew number of relevant depenant minors
                                // this adult has
        int rdc_teens;          // The number of relevant dependant teens
                                // this adult has
        int mc_minors;          // The number of multi-case minors not of
                                // this case
        int mc_teens;           // The number of multi-case teens not of
                                // this case
        int ati;                // The Adjusted Taxable Income for this
                                // adult
        int ssa;                // The Self-Support Amount
        int ati_less_ssa;       // The ATI less the SSA
        int rdc_cost;           // The cost of the relevant dependant
                                // children
        int ati_less_rdc_cost;  // The ATI less the rdc cost
                                // (also less the SSA)
        int mca;                // The multi-case allowance
        int mccap;              // The Muli-case cap
        int ati_less_mca;       // The ati less the mca (also les the rdc
                                // cost and the SSA)
        int csi;                // The child support income of this adult
        float pip;              // The parents income support percentage
        float pays;             // The amount this adult pays in CS
        float gets;             // The amount this adult receives in CS
        bool splitcase;         // Whether or not this is a split case
        std::string Display();
        std::deque<cs_full_relationship> rlt;   // The relationships to the
                                                //CS children in this case.
        std::deque<cs_full_relationship> teenmc;    // List of Cost
                                           //percentages for the multi-case teens.
        std::deque<cs_full_relationship> minormc;   // List of cost percentages
                                                    // for the multi-case teens.
};
//===========================================================================
// Class for a full case
// A full case an the overall case, an instance is built for all child cases
// that have the same parents.
//===========================================================================
class cs_fullcase {
    public:
        cs_fullcase(int);
        ~cs_fullcase();
        void SetCostPerChild(float);
        float GetCostPerChild();
```

```
        void AddChildCase(int);
        int GetfccaseSize();
        int Getfccase(int);
        std::string GetfccaseChildName(int);
        void SetChildCounts(int, int);
        int GetfacaseSize();
        int GetfacaseAdult(int);
        void AddFullAdultCase(int, std::string, int, int);
        std::string GetfacaseAdultName(int);
        int GetfacaseATI(int);
        int GetfacaseSSA(int);
        int GetfacaseATILessSSA(int);
        int GetfacaseRDCCost(int);
        int GetfacaseATILessRDCCost(int);
        int GetfacaseRDCMinors(int);
        int GetfacaseRDCTeens(int);
        int GetfacaseMCMinors(int);
        int GetfacaseMCTeens(int);
        float GetfacasePIP(int);
        void SetfacasePIP(int, float);
        void AddFullRelationshipToAdult(int, int,std::string, int, int,
            int, int, int, bool);
        void AddMCTeenRltToAdult(int, int, std::string, int, int, int,
            int, int);
        void AddMCMinorRltToAdult(int, int, std::string, int, int, int,
            int, int);
        int GetAdultsRltSize(int);
        int GetAdultsMCMinorsSize(int);
        int GetAdultsMCTeensSize(int);
        int GetAdultsMCMinorsCostPercentage(int, int);
        int GetAdultsMCTeensCostPercentage(int, int);
        std::string GetAdultsRltChildsName(int, int);
        std::string GetAdultsMCTeensChildsName(int, int);
        std::string GetAdultsMCMinorsChildsName(int, int);
        int GetAdultsRltCare(int, int);
        int GetAdultsRltAdult(int, int);
        int GetAdultsRltChild(int, int);
        int GetAdultsRltRType(int, int);
        int GetAdultsRltCarePercentage(int, int);
        int GetAdultsRltCostPercentage(int, int);
        int GetAdultsRltPreAdjCostOfTheChild(int, int);
        int GetAdultsRltCostOfTheChild(int, int);
        int GetAdultsRltMultiCaseCap(int, int);
        int GetAdultsRltMultiCaseCost(int, int);
        bool GetAdultsRltAge(int, int);
        void AddRelevantDependantMinor(int);
        void AddRelevantDependantTeen(int);
        void AddMultiCaseMinor(int);
        void AddMultiCaseTeen(int);
        void SetfacaseRDCCost(int, int);
        void SetAdultsMCMinorsValues(int, int, int, int);
        void SetAdultsMCTeensValues(int, int, int, int);
        void SetfacaseMCCost(int, int);
        int GetfacaseMCA(int);
        int GetfacaseATILessMCA(int);
        int GetfacaseCSI(int);
        int GetfacaseMCCap(int);
        void IncreaseCombinedCSIncome(int amnt, cs_amounts);
        int GetCCSI();
        int GetCOC();
        void SetParentAbroad();
        void SetNonParentCarer();
        void SetAdultsRltMCCap(int, int, int);
        void SetAdultsRltPCSPC(int, int, float);
        float GetAdultsRltPCSPC(int, int);
        void SetAdultsRltPreAdjCOTC(int, int, int);
        void SetAdultsRltCOTC(int, int, int);
        int GetAdultsMCMinorsMCCost(int, int);
        int GetAdultsMCTeensMCCost(int, int);
        std::string AdultsMCTeensDisplay(int, int);
        std::string AdultsMCMinorsDisplay(int, int);
```

```
        bool IsParentAbroadApplicable();
        void AddfacaseCSCost(int, float);
        float GetfacasePays(int);
        void AddfacaseCSGets(int, float);
        float GetfacaseCSGets(int);
        void AddAdultsRltRcvTotCare(int, int, int);
        int GetAdultsRltRcvTotCare(int, int);
        void SetfacaseMCCap(int, int);
        std::string Display();
        friend class cs_full_adultcase;
        friend class cs_full_relationship;
        std::deque<int> fccase;
        std::deque<cs_full_adultcase> facase;
    private:
        bool splitcare;
        int ccsi;                           //combined CS income
        bool parent_abroad;
        bool non_parent_carer;
        int coc;                            // overall cost of children
        float costperchild;
        int cs_teens;                       // number of CS teenagers
        int cs_minors;                      // number of CS minors
};
```