EXPLICIT FINAL REVIEW

1. **Consider the following class declarations when answering this question:**

**(IMPLEMENT THE COPY CONSTRUCTOR TO PERFORM A DEEP COPY)**

class address\_record       class address\_book

{                                         {

    public:                               public:

        string name;                          address\_book(const address\_book &);

        string address;                       .......

        string phone;   private:

int miles\_away; int count;//cells in used

};  int capacity;//tot cells

                                                   address\_record \*address\_DB; // dyn array

                                          };

Implement the copy constructor for the class address\_book.

1. Consider the following class declarations when answering this question:

**(FINDING THE SUM OF A FIELD IN THE RECORDS IN A DYNAMIC ARRAY)**

class address\_record       class address\_book

{                                         {

    public:                               public:

        string name;                          int closest( const int threshold);

        string address;                       .......

        string phone;   private:

int miles\_away; int count;//cells in used

};  int capacity;//tot cells

                                                   address\_record \*address\_DB; // dyn array

                                          };

Implement the function "closest". The function will  return the sum of all the address\_records stored in address\_DB with the “miles\_away” field greater than or equal to “ threshold”. See the prototype for "closest" inside the class declaration for the class "address\_book".

1. **Consider the following class declarations when answering this question:**

**(FRIEND OPERATOR OVERLOADING WITH CHAINING)**

class address\_record       class address\_book

{                                         {

    public:                               public:

        string name;                          friend ostream & operator<<( ostream &, const address\_book & org);

        string address;                       .......

        string phone;   private:

int miles\_away; int count;//cells in used

};  int capacity;//tot cells

                                                   address\_record \*address\_DB; // dyn array

                                          };

Implement the overloaded "operator<<" with chaining. This function will print all the fields of every address\_record stored in address\_DB to the screen.

1. **Consider the following class declarations when answering this question:**

**(OPERATOR OVERLOADING AS A MEMBER FUNCTION WITHOUT CHAINING –**

**ALSO EXAMPLE OF ADDING TO THE END OF AN ARRAY)**

class address\_record       class address\_book

{                                         {

    public:                               public:

        string name;                          void operator+( const address\_record &);

        string address;                       .......

        string phone;   private:

int miles\_away; int count;//cells in used

};  int capacity;//tot cells

                                                   address\_record \*address\_DB; // dyn array

                                          };

Implement the overloaded "operator+" without chaining as a member function. This function will add an address\_record to address\_DB only if the name field does not match any of the address\_records stored in address\_DB. If a name matches any record in address\_DB, do not add it, and print the message "duplicate record".  If address\_DB is full print the message "address\_DB is full".

1. **Consider the following class declarations when answering this question:**

**( DELETE A RECORD FROM A DYNAMIC ARRAY)**

class address\_record       class address\_book

{                                         {

    public:                               public:

        string name;                          void delete( const string key &);

        string address;                       int search( const string & key).

        string phone;   private:

int miles\_away; int count;//cells in used

};  int capacity;//tot cells

                                                   address\_record \*address\_DB; // dyn array

                                          };

Implement the function "delete" which removes the address\_record with a name field that matches key. You may use the search function to help you implement this function. Assume search returns -1 if key is not in address\_DB; otherwise it returns the location of the address\_record that matches key.