**BLG 233E**

**DATA STRUCTURES AND LABORATORY**

CRN: 11146

**REPORT OF HOMEWORK #3**

Submission Date: 25.12.12

**STUDENT NAME: TUĞRUL YATAĞAN**

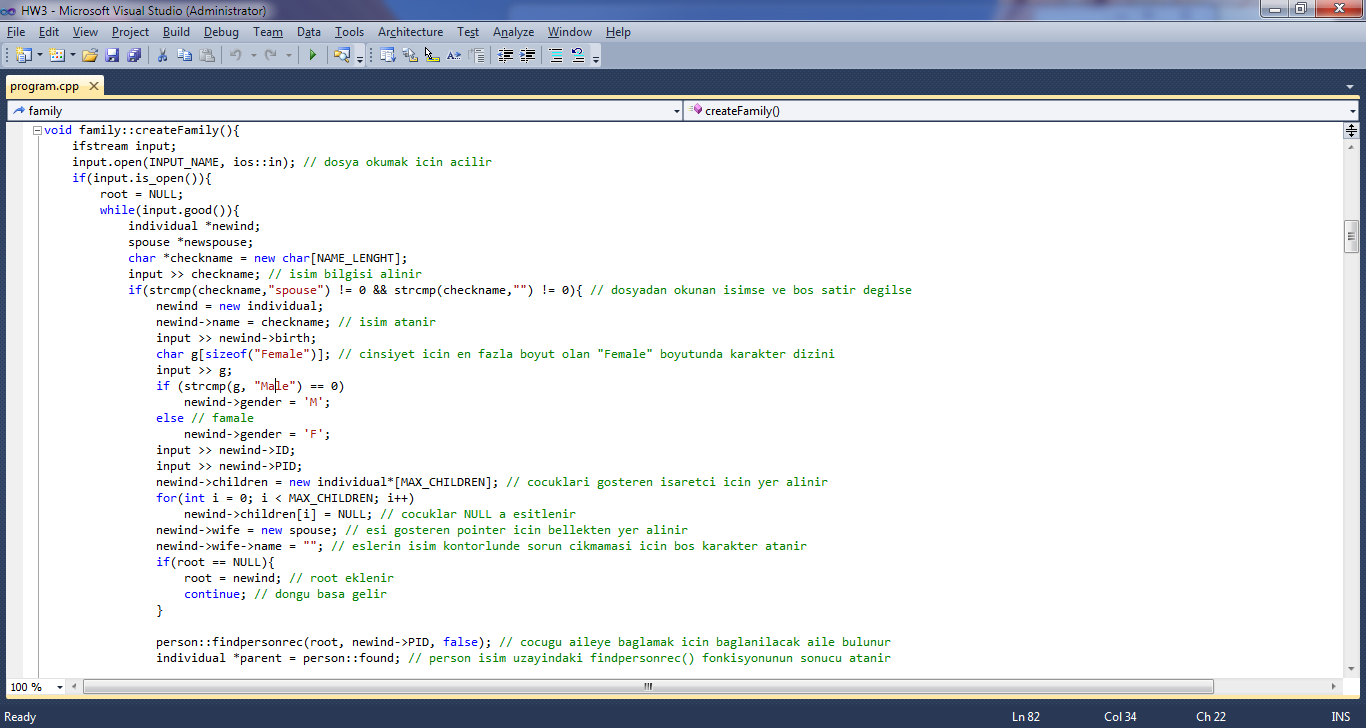
**STUDENT NUMBER: 040100117**

**1. Introduction**

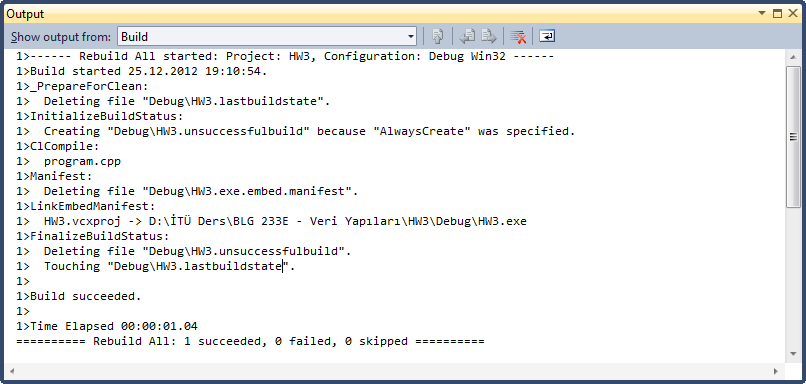
In this homework, a family was represented using a tree. Each individual was represented as a node in the tree, and relationships will be represented with the connections between the nodes. There are two types of people, namely; individual and spouse.

**2. Development and Operating Environments**

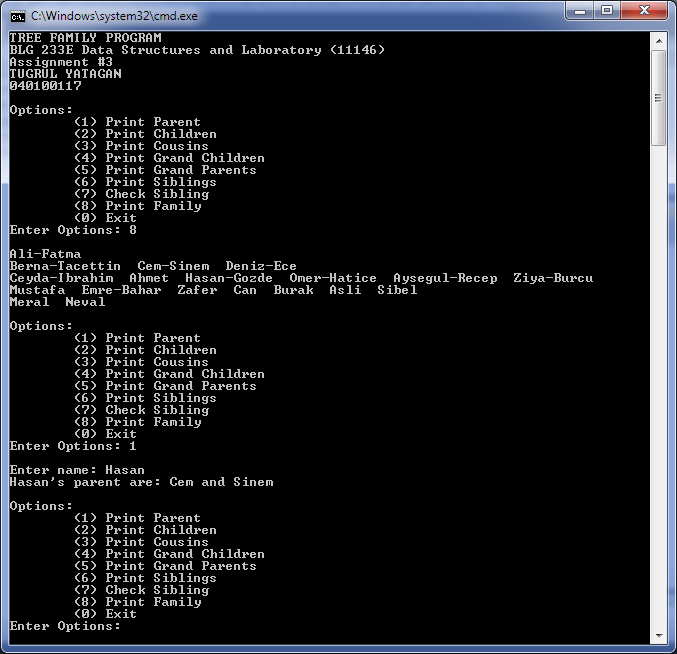
Microsoft Visual C++ 2010 environment has been used to write the source code in Windows 7 operation system and again Microsoft Visual C++ 2010 compiler was used to compile the program.



The program compiled without warning or error:



Finally the program is executed. Sample outcome is below:



**3. Data Structures and Variables**

Each individual have a name (char\*), year of birth (int), gender (char), pointer array to children (node\*\*) pointer to wife/husband (spouse\*), node ID (int), and parent ID (int). The spouse type has two data types: name (char\*) and year of birth (int).

struct spouse{

char \*name;

int birth;

};

struct individual{

char \*name;

int birth;

char gender;

individual \*\*children;

spouse \*wife;

int ID;

int PID;

};

- createFamily(): This function creates the tree from input text file. Individual and spouse structs was added according to relatives when the program starts.

- printParent(individual \*traverse, char \*search, bool stop): This function prints the names of the parents (both mother and father) for the given parameter.

- printChildren(individual \*traverse, char \*search, bool stop): This function prints the names of the children for the given parameter.

- printCousins(individual \*traverse, char \*search, bool stop): This function should print the names of the cousins for the given parameter. Here, cousins are only the people who have the same grandparents.

- printGrandchildren(individual \*traverse, char \*search, bool stop): This function should print the names of the grandchildren for the given parameter.

- printGrandparents(individual \*traverse, char \*search, bool stop): This function should print the names of the grandparents for the given parameter.

- printSiblings(individual \*traverse, char \*search, bool stop): This function prints the names of the siblings for the given parameter.

- isSibling(char \*ptr1, char \*ptr2): This function returns true if the two parameters have the same parents.

-printFamily(): This function prints all the tree layer by layer.

-closeFamily(individual \*traverse): This function deletes the tree.

namespace person{

bool findpersonrec(individual \*, int, bool);

individual \*found;

}

namespace sibling{

bool checkParent(individual \*,char \*, bool);

individual \*result;

}

namespace level{

int findLayerNum(individual \*, int);

int printLayer(individual \*, int, int);

int max\_layer;

}

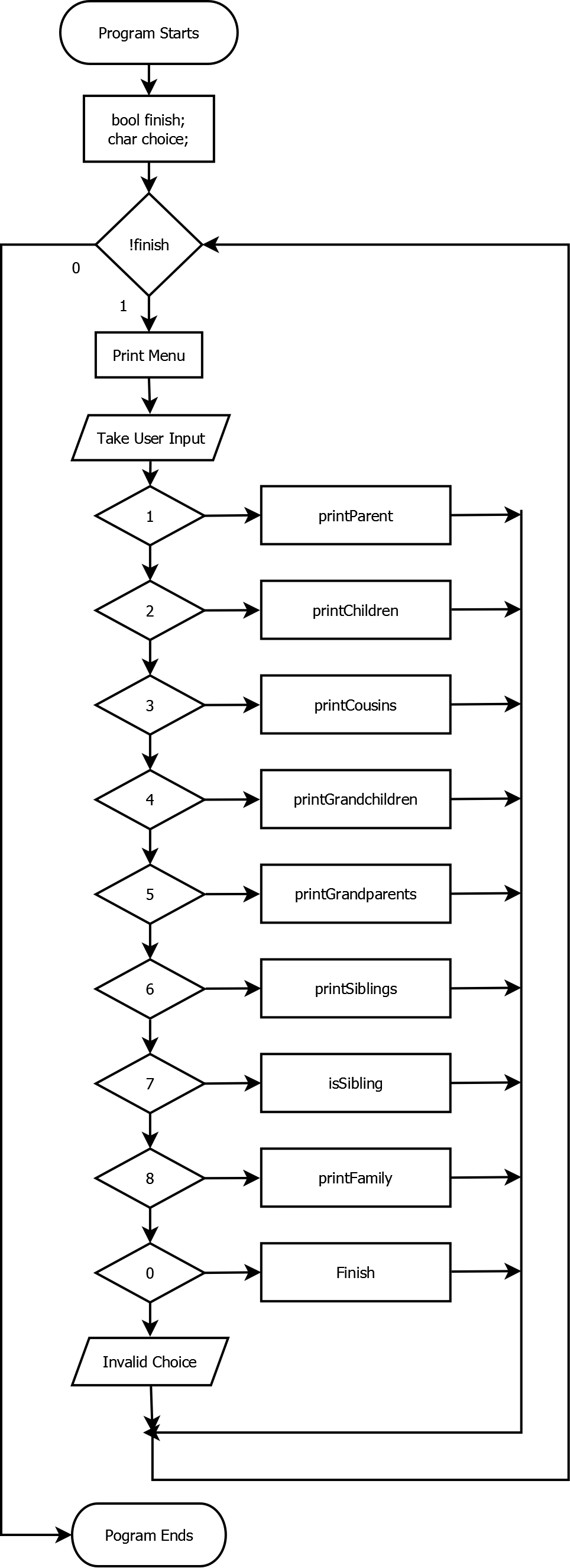
-findpersonrec(individual \*, int, bool): This function finds the persons position for creating the tree. Return value is stored in individual \*found variable.

checkParent(individual \*,char \*, bool): This function finds the parents of parameter for sibling check. Return value is stored in individual \*result variable.

findLayerNum(individual \*, int): This function counts the depth of the tree (total number of layer) for printing the tree. Return value is stored in int max\_layer variable.

printLayer(individual \*, int, int): This function prints wanted layer of the tree.

**6. Program Flow**



**5. Conclusion**

In this homework, I have become more familiar with the concept of data structures, trees and recursive functions. I had the chance to intensify my knowledge about their structures.