CSC 17a C++:Objects **Name:** Daniel Baca 12/14/2022
Fall 2022
Final
12/07/22
Time Limit: Maximum of 7 days

The Midterm contains 9 pages and 7 questions. Total points possible are 100.

Submit the Final to the canvas assignment marked Final. The solutions/programs should be in a folder marked final and the contents zipped. Use Netbeans, delete the build and dist folders before zipping. Do the best you can and turn in as much as you can.

I just want one program for the solution to the problems that follow. The program should prompt the user for which problem solution to display. Use a do-while and switch construct like the included menu program.

Develop each in a separate project. Then combine into one project when done. You are to include each project as well as the combined menu project when submitting your final.

```
Addendum: Extra Credit  (Choose several for no more than 10 pts)
Alert me to any extra credit in the output of the problem.
1)  5   pts->Modify the toString() function in Problem 4 or 5 to return a
        string or for the most credit a char *;
2)  5   pts->Utilize problem 4 or 5 and Serialize the object to a binary
        file.  Prove that it was done correctly.
        Get even more credit by serializing Problem 3.
3)  2.5 pts->Utilize problem 3 to overload the + operator.  Then take
        2 arrays and add them together from this class.
4)  2.5 pts->Convert Problem 1 by using templates in place of the character
        arrays.
5)  2.5 pts->Exception handling for Problems 4 and 5
6)  10  pts->Do problem 4 from the programming competition in folder, Use a
        class that can be called to perform the sort.  You might want to
        look at problem 2 and maybe kill 2 birds with one stone! :)
7)  1   pt->Discuss any code that you would make better and support your
        reasoning.
8)  5   pts->Utilize regular expressions on all problems requiring inputs.
```

---

1. (15 points) Problem 1 (Random Sequence) Create a class that returns a random number from the following set, 19,34,57,79,126. Loop 100,000 times with this procedure and print the frequency of each of the 5 numbers obtained. The following is the specification for the class.

   Specification

   ```
   class Prob1Random{
       private:
           char *set;      //The set of numbers to draw random numbers from
   ```

```
        char  nset;     //The number of variables in the sequence
        int  *freq;     //Frequency of all the random numbers returned
        int   numRand;  //The total number of times the random number
                        //function is called
    public:
        Prob1Random(const char,const char *);//Constructor
        ~Prob1Random(void);                  //Destructor
        char randFromSet(void);              //Returns a random number from the set
        int *getFreq(void) const;            //Returns the frequency histogram
        char *getSet(void) const;            //Returns the set used
        int getNumRand(void) const;          //Gets the number of times randFromSet
                                             //has been called
};

Driver program to return a random sequence
    char n=5;
    char rndseq[]={19,34,57,79,126};
    int ntimes=100000;
    Prob1Random a(n,rndseq);
    for(int i=1;i<=ntimes;i++){
        a.randFromSet();
    }
    int *x=a.getFreq();
    char *y=a.getSet();
    for(int i=0;i<n;i++){
        cout<<int(y[i])<<" occurred "<<x[i]<<" times"<<endl;
    }
    cout<<"The total number of random numbers is "<<a.getNumRand()<<endl;
```

Sample Output

```
 19  occurred 20045 times
 34  occurred 19952 times
 57  occurred 20035 times
 79  occurred 20039 times
126  occurred 19929 times
```

The total number of random numbers is 100000

Note: Your results are not expected to be exactly the same! After all these are pseudo-random number sequences with different seeds.

2. (15 points) Problem 2 (All Kinds of Sorting) Sort a single column array and/or sort a 2 dimensional array of characters given any column. Here is what I used as my template specification.

```
//This class sorts arrays either ascending or descending
template<class T>
class Prob2Sort{
    private:
        int *index;                                 //Index that is utilized
                                                    //in the sort
    public:
        Prob2Sort(){index=NULL;};                   //Constructor
        ~Prob2Sort(){delete []index;};              //Destructor
        T * sortArray(const T*,int,bool);           //Sorts a single column array
        T * sortArray(const T*,int,int,int,bool);//Sorts a 2 dimensional array
                                                    //represented as a 1 dim array
};
```

Driver program for the above class. Create your own file to read.

```
    cout<<"The start of Problem 2, the sorting problem"<<endl;
    Prob2Sort<char> rc;
    bool ascending=true;
    ifstream infile;
    infile.open("Problem2.txt",ios::in);
    char *ch2=new char[10*16];
    char *ch2p=ch2;
    while(infile.get(*ch2)){cout<<*ch2;ch2++;}
    infile.close();
    cout<<endl;
    cout<<"Sorting on which column"<<endl;
    int column;
    cin>>column;
    char *zc=rc.sortArray(ch2p,10,16,column,ascending);
    for(int i=0;i<10;i++){
        for(int j=0;j<16;j++){
            cout<<zc[i*16+j];
        }
    }
    delete []zc;
    cout<<endl;
```

The output from this problem.

```
The start of Problem 2, the sorting problem
Lbekoeddhoffbmg
Lkcmggjcdhhglif
Cgldjhcekjigcdd
Cgldjhcekjigcdo
```

```
Bffmdbkcenlafjk
Fggdijijegfblln
Jjlncnimjldfedj
Amliglfohajcdmn
Balgfcaelhfkgea
Kmlhmhcddfoeilc

Sorting on column 15
Cgldjhcekjigcdo
Fggdijijegfblln
Amliglfohajcdmn
Bffmdbkcenlafjk
Jjlncnimjldfedj
Lbekoeddhoffbmg
Lkcmggjcdhhglif
Cgldjhcekjigcdd
Kmlhmhcddfoeilc
Balgfcaelhfkgea
```

3. (15 points) Problem 3 (Spreadsheet Stuff)

```
Class Specifications

template<class T>
class Prob3Table{
    protected:
        int rows;                                   //Number of rows in the table
        int cols;                                   //Number of cols in the table
        T *rowSum;                                  //RowSum array
        T *colSum;                                  //ColSum array
        T *table;                                   //Table array
        T grandTotal;                               //Grand total
        void calcTable(void);                       //Calculate all the sums
    public:
        Prob3Table(char *,int,int);                 //Constructor then Destructor
        ~Prob3Table(){delete [] table;delete [] rowSum;delete [] colSum;};
        const T *getTable(void){return table;};
        const T *getRowSum(void){return rowSum;};
        const T *getColSum(void){return colSum;};
        T getGrandTotal(void){return grandTotal;};
};

template<class T>
class Prob3TableInherited:public Prob3Table<T>{
```

```
    protected:
        T *augTable;                                    //Augmented Table with sums
    public:
        Prob3TableInherited(char *,int,int);           //Constructor
        ~Prob3TableInherited(){delete [] augTable;};  //Destructor
        const T *getAugTable(void){return augTable;};
};

Driver code

    cout<<"Entering problem number 3"<<endl;
    int rows=5;
    int cols=6;
    Prob3TableInherited<int> tab("Problem3.txt",rows,cols);
    const int *naugT=tab.getTable();
    for(int i=0;i<rows;i++){
        for(int j=0;j<cols;j++){
            cout<<naugT[i*cols+j]<<" ";
        }
        cout<<endl;
    }
    cout<<endl;
    const int *augT=tab.getAugTable();
    for(int i=0;i<=rows;i++){
        for(int j=0;j<=cols;j++){
            cout<<augT[i*(cols+1)+j]<<" ";
        }
        cout<<endl;
    }


Example Input Table
   101   101   102   103   104   105
   106   107   108   109   110   111
   112   113   114   115   116   117
   118   119   120   121   122   123
   124   125   126   127   128   129
Example Output Table with rows summed, columns summed, and the grand total printed.
   101   101   102   103   104   105   616
   106   107   108   109   110   111   651
   112   113   114   115   116   117   687
   118   119   120   121   122   123   723
   124   125   126   127   128   129   759
   561   565   570   575   580   585  3436
```

4. (15 points) Problem 4 (Savings Account Class) Create a SavingsAccount class with the following specification

```
public:
    SavingsAccount(float);                //Constructor
    void  Transaction(float);             //Procedure
    float Total(float=0,int=0);           //Savings Procedure
    float TotalRecursive(float=0,int=0);
    void  toString();                     //Output Properties
private:
    float Withdraw(float);                //Utility Procedure
    float Deposit(float);                 //Utility Procedure
    float Balance;                        //Property
    int   FreqWithDraw;                   //Property
    int   FreqDeposit;                    //Property
```

1) The constructor initilizes the balance if greater than 0 and sets the other properties to 0.
2) If the transaction is greater than 0 then a Deposit is made else a Withdraw is made.
3) The balance is increased with a deposit but decreased if a Withdrawal. This assumes the Withdrawal is less than the balance. Can't have a negative balance. Tell the user that he is trying to make a withdrawal that exceeds his balance.
4) When a WithDrawal is made increment FreqWithDraw else if a Deposit is made increment FreqDeposit.
5) The toString procedure outputs all properties.
6) The total procedure tells you how much you will have in savings given the interest rate and the amount of time. Total(float savint,int time) returns Balance*(1+savint)^time. Utilize a for loop for this calculation.
7) See if you can write a recursive procedure that does the same thing as 6). Call it TotalRecursive.
8) Think of what follows as pseudocode. The random number generator below produces a number between 0 and 32,767. If you fashion a random number that will do the same then you will get positive and negative transactions (-500,500). The output simply calculates the current balance with a 10 percent interest rate and 7 years worth of compounding. Also, you tried to start out with a negative balance which should have been initialized to 0.

Driver code

```
SavingsAccount mine(-300);
```

```
    for(int i=1;i<=10;i++){
        mine.Transaction((float)(rand()%500)*(rand()%3-1));
    }
    mine.toString();
    cout<<"Balance after 7 years given 10% interest = "
        <<mine.Total((float)(0.10),7)<<endl;
    cout<<"Balance after 7 years given 10% interest = "
        <<mine.TotalRecursive((float)(0.10),7)
        <<" Recursive Calculation "<<endl;
```

```
Example Output:
    WithDraw not Allowed
    WithDraw not Allowed
    Balance=1056
    WithDraws=4
    Deposit=5
    Balance after 7 years given 10% interest = 2057.85
    Balance after 7 years given 10% interest = 2057.85 Recursive Calculation
```

5. (15 points) Problem 5 (Employee Class) Create an Employee class with the following specification

```
    public:
        Employee(char[],char[],float);  //Constructor
        float  CalculatePay(float,int); //Procedure
        float  getGrossPay(float,int);  //Procedure
        float  getNetPay(float);        //Procedure
        void   toString();              //Procedure
        int    setHoursWorked(int);     //Procedure
        float  setHourlyRate(float);    //Procedure
    private:
        double Tax(float);      //Utility Procedure
        char   MyName[20];      //Property
        char   JobTitle[20];    //Property
        float  HourlyRate;      //Property
        int    HoursWorked;     //Property
        float  GrossPay;        //Property
        float  NetPay;          //Property
```

```
1)  The constructor inputs the Name, Job Title and Hourly rate of the employee.
2)  All other properties are initialized to zero in the constructor.
3)  The Tax utility routine calculates the tax based on
        .1 for GrossPay < 500,
```

```
        .2 for GrossPay<1000 but greater than 500, and
        .3 for anything above 1000.
4)  The Set procedures simply write and return the obvious properties.
5)  The toString procedure prints all the properties.
6)  The getGrossPay procedure calculates straight time for hours worked < 40,
        time and 1/2 for hours worked < 50 but greater than 40, double time for
        all hours above 50.  The inputs to this routine are the hourly rate and
        the hours worked.
7)  The net pay routine returns the gross pay subtracting off any taxes.
        Realize that the Tax procedure is utilized but these taxes are
        progressive rates for income above the cutoff limit as described in 3).
        Taxes are calculated for 10% below 500, 20% for amounts between 500
        and 1000 and finally anything above 1000 is calculated at the 30% rate.
8)  The Hoursworked must be > 0 and < 84
9)  The Hourlyrate must be > 0 and < 200
10) The calculate pay routine returns the net pay.
        return getNetPay(getGrossPay(setHourlyRate(x),setHoursWorked(y)));

Example Input
    Employee Mark("Mark","Boss",215.50);
    Mark.setHoursWorked(-3);
    Mark.toString();
    Mark.CalculatePay(Mark.setHourlyRate(20.0),Mark.setHoursWorked(25));
    Mark.toString();
    Mark.CalculatePay(Mark.setHourlyRate(40.0),Mark.setHoursWorked(25));
    Mark.toString();
    Mark.CalculatePay(Mark.setHourlyRate(60.0),Mark.setHoursWorked(25));
    Mark.toString();

    Employee Mary("Mary","VP",50.0);
    Mary.toString();
    Mary.CalculatePay(Mary.setHourlyRate(50.0),Mary.setHoursWorked(40));
    Mary.toString();
    Mary.CalculatePay(Mary.setHourlyRate(50.0),Mary.setHoursWorked(50));
    Mary.toString();
    Mary.CalculatePay(Mary.setHourlyRate(50.0),Mary.setHoursWorked(60));
    Mary.toString();

Example Output
Unacceptable Hourly Rate
Unacceptable Hours Worked
Name = Mark Job Title = Boss
 Hourly Rate = 0 Hours Worked = 0 Gross Pay = 0 Net Pay = 0
Name = Mark Job Title = Boss
 Hourly Rate = 20 Hours Worked = 25 Gross Pay = 500 Net Pay = 450
```

```
Name = Mark Job Title = Boss
 Hourly Rate = 40 Hours Worked = 25 Gross Pay = 1000 Net Pay = 850
Name = Mark Job Title = Boss
 Hourly Rate = 60 Hours Worked = 25 Gross Pay = 1500 Net Pay = 1200
Name = Mary Job Title = VP
 Hourly Rate = 50 Hours Worked = 0 Gross Pay = 0 Net Pay = 0
Name = Mary Job Title = VP
 Hourly Rate = 50 Hours Worked = 40 Gross Pay = 2000 Net Pay = 1550
Name = Mary Job Title = VP
 Hourly Rate = 50 Hours Worked = 50 Gross Pay = 2750 Net Pay = 2075
Name = Mary Job Title = VP
 Hourly Rate = 50 Hours Worked = 60 Gross Pay = 3750 Net Pay = 2775
```

6. (15 points) Problem 6 (Conversions)

```
Given the following base 10 decimals
a)5.75
b)0.9
c)99.7
1) Convert to binary, octal and hex, then
2) Convert to NASA Hex float with first 24 bits representing the
signed fraction and the last 8 bits representing the
signed exponent. Scaled as 0.FRACTION x 2^EXPONENT
3) convert a) to scaled integer binary 1 unsigned byte maximum bits,
   convert b) to scaled integer binary 2 unsigned byte maximum bits,
   convert c) to scaled integer binary 3 unsigned byte maximum bits,
4) multiply 3) by 1 byte value 7 then shift back to integer and
        output the result
5) Then convert using IEEE 754 format
```

7. (10 points) Menu