

CMPS 150
Sample Final Exam
SOLUTION

1. Write a **void** function that is passed an **int** array and a single **int**. The single **int** parameter is the number of elements in the **int** array parameter. The function will find the smallest value in the array and it will compute the average of the array values. Both items will be printed with a meaningful message.

```
void Process(int array[], int n)
{
    double sum, average;
    int i, small;

    sum = 0.0;
    small = array[0];
    sum = sum + array[0];

    for (i=1; i<n; i++)
    {
        if (array[i] < small)
            small = array[i];

        sum = sum + array[i];
    }

    average = sum / n;
    cout<<"Smallest number is: "<<small<<endl;
    cout<<"Average number is: "<<average<<endl;
}
```

2. Write a value-returning function that takes a single **ifstream** as an input parameter (by reference) and **returns** a single **int**. The single **int** that is returned is the number of items read from the input file parameter. The input file has been opened successfully and contains an unknown number of lines of text. Read each word from the input file until **EOF**, storing each word in a **string** array. You may assume there are no more than 500 words in the file. Print out the array of words – in REVERSE order – BUT, only print those words that begin with the lowercase letter 'k'. The test for this is:
`if (stringVariableName.substr(0,1) == 'k')`

```
int Process(ifstream& inFile)
{
    int    i, count;
    string words[500];

    count = 0;

    inFile>>words[count];

    while(inFile)
    {
        count++;
        inFile>>words[count];
    }

    for (i=count-1; i>=0; i--)
    {
        if (words[i].substr(0,1) == 'k')
            cout<<words[i]<<endl;
    }

    return count;
}
```

3. Write ONLY the **main** function for a program that offers the user a menu of three calculation options: Monthly Loan Payments, Yearly Interest Accrued, Amortization Schedule. There should be a fourth menu item to quit. Write the code to display the menu and a **switch** statement to call an appropriate function. The functions called will be passed NO parameters, and will all return a **double** using a return statement. For example, to call the function to compute yearly interest accrued could be: **yearlyInterest = ComputeYrInt();**
All invalid menu selections should respond with an error and re-display the menu.

```
void main()
{
    double  monthlyPay, yearlyInterest, amortization;
    int     choice;

    do
    {
        cout<<"1. Monthly Loan Payments\n";
        cout<<"2. Yearly Interest Accrued\n";
        cout<<"3. Amortization Schedule\n";
        cout<<"4. Quit\n";

        cout<<"Choice: ";
        cin>>choice;

        switch(choice)
        {
            case 1:  monthlyPay = DoMonthlyPayment();
                    break;
            case 2:  yearlyInterest = DoYearlyInterest();
                    break;
            case 3:  amortization = DoAmortization();
                    break;
            case 4:  break;
            default: cout<<"Invalid Choice - try again !\n";
                    break;
        }
    } while (choice != 4);
}
```

4. Write a **struct** statement to create a structured data type called **Vehicle** that can store the following data: model, make, year, price. Then declare a variable of type **Vehicle** and set each data member to information about your favorite vehicle.

```
struct Vehicle
{
    string model;
    string make;
    int    year;    // they may use a string instead
    double price;
};
```

```
Vehicle myVehicle = {"Land Cruiser", "Toyota", 2002, 52780.99};
```

OR

```
Vehicle myVehicle;
```

```
myVehicle.model = "Land Cruiser";
myVehicle.make  = "Toyota";
myVehicle.year  = 2002;
myVehicle.price = 52780.99;
```

5. Write the specification for a class **Hardware** which has five private data members, nine public member functions and two public constructor functions. The data members are:

inventory code	(string)
description	(string)
quantity on hand	(integer)
unit price	(double)
is it on sale	(bool)

The functions are:

QtyCheck	checks availability, passed nothing, returns the quantity data member
CheckOut	buy an item, passed an integer, decrements the quantity data member by the parameter, returns nothing
CalcCost	computes total cost, passed an integer indicating number purchased, returns the unit cost * int parameter
Write	passed nothing, prints hardware info “nicely”, returns nothing
GetDescr	passed nothing, returns the description data member
GetPrice	passed nothing, returns the unit price data member
GetCode	passed nothing, returns the inventory code data member
GetOnSale	passed nothing, returns whether on the item is on sale
UpdateQty	passed an integer, updates the quantity data member to the value of the parameter passed, returns nothing

parameterized constructor - passed (string,string,int,double,bool), sets each data member to parameters passed

default constructor - passed nothing, sets string data members to blanks, int to 0, double to 0.0, bool to false

```
class Hardware
{
    private:
        string  code;
        string  description;
        int     qty;
        double  price;
        bool    onSale;

    public:
        int     QtyCheck();
        void    CheckOut(int);
        double  CalcCost(int);
        void    Write();
        string  GetDescr();
        double  GetPrice();
        string  GetCode();
        bool    GetOnSale();
        void    UpdateQty(int);
        Hardware(string, string, int, double, bool);
        Hardware();
};
```

6. Implement: **CheckOut**, **CalcCost**, **GetPrice**, **UpdateQty** and **parameterized constructor** functions of #5.

```
void Hardware::CheckOut(int numBought)
{
    qty = qty - numBought;
}

double Hardware::CalcCost(int numBought)
{
    return price * numBought;
}

double Hardware::GetPrice()
{
    return price;
}

void Hardware::UpdateQty(int newQty)
{
    qty = newQty;
}

Hardware::Hardware(string a,string b,int c,double d,bool e)
{
    code = a;
    description = b;
    qty = c;
    price = d;
    onSale = e;
}
```

7. Write the client code to declare an array of 75 **Hardware** objects.

```
Hardware myHw[75];
```

8. Write the client code to read data into the objects of #7, from a file of the user's choice.

```
cout<<"Enter input file name: ";
cin>>filename;

inFile.open(filename.c_str());

count = 0;

inFile>>a>>b>>c>>d>>e;

while(inFile)
{
    myHw[count] = Hardware(a,b,c,d,e);
    inFile>>a>>b>>c>>d>>e;
}
```

9. Write the client code to buy (check out) eight(8) of the item in the 14th element (be careful) of the array, print out that item's info, and print out its total cost.

```
myHw[13].Checkout(8);

myHw[13].Write();

cost = myHw[13].CalcCost(8);

cout<<"Total Cost: "<<cost<<endl;
```

Name: _____

Section: _____

10. Show the array elements each time you iterate the outer for loop of this sort routine. NOTE: $n = 6$
SHOW YOUR WORK ON THIS PAGE

```
void Sort(int list[], int n)
{
    int i, j, temp;
    int min, indexSmallest;

    for(i=0; i<n-1; i++)
    {
        indexSmallest = i;
        min = array[i];

        for(j=i+1; j<n; j++)
        {
            if (array[j]<min)
            {
                min = array[j];
                indexSmallest = j;
            }
        }

        temp = array[i];
        array[i] = array[indexSmallest];
        array[indexSmallest] = temp;

        //****
        //**** Fill in next column of array elements with current values
        //****

    } // end of the Sort function
```

5	4	4	4	4	4
31	31	5	5	5	5
13	13	13	13	13	13
16	16	16	16	16	16
90	90	90	90	90	31
4	5	31	31	31	90