

Dr. Rodger

String class

```
int length();
// post: returns number of characters in string

string substr(int pos, int len);
// pre: 0 <= pos < length()
// post: returns substring of len chars, starting at pos
//       (as many characters as possible if len too large)

int find(string s)
// post: returns first position at which s begins
//       (returns string::npos if s does not occur)
```

Dice class

```
class Dice{
public:
    Dice(int sides);    // constructor, sides specifies number of sides
    int Roll();        // return the random roll between 1 and num. of sides
    int NumSides();    // returns how many sides this die has
    int NumRolls();    // returns num. of times this die rolled
private:
    // not shown
};
```

PROBLEM 1 : (*Output:* (14 pts))

Give the output of the following statements.

```
cout << 43 << "*" << "78" << endl;

int number = 21;
double y = 14.0;
cout << number << "*" << number/5 << "*" << number/y << endl;
cout << number % 5 << endl;

string word = "ball";
string phrase = "basketball starts soon, throw the ball";
cout << word.length() << endl;
cout << phrase.find(word) << endl;
int pos = phrase.find(word);
cout << phrase.substr(pos-3, 2) << endl;
cout << word + word << endl;
```

Output

PROBLEM 2 : (*More Output: (10 pts)*)

Give the output of the following statements based on the users input.

```
int number;
cout << "enter number: ";
cin >> number;

if (number > 5)
    cout << "A";
else if (number > 9)
    cout << "B";
else
    cout << "C";

do
{
    cout << "HA";
    number --;
} while (number > 8);

cout << endl;
```

What is the output if the user enters 7?

What is the output if the user enters 9?

PROBLEM 3 : (*Generating Random Names: (28 pts)*)

In this problem, you will generate the names of two people who will be partners for CPS 6.

PART A (*8 pts*):

Write the function *ReadNames* whose header is given below. *ReadNames* takes a parameter *input*, an input file stream that has already been bound to a file and is open for reading, and reads all the names from the stream placing them into the reference parameter *allnames*, with a ":" after each name. In addition, the reference parameter *count* is assigned the number of names read in.

For example, if *input* contained the names:

```
Avila
Cutrona
Green
Hahn
```

Then after the call `ReadNames(input, names, number)`, *number* would be 4 and *names* would be "Avila:Cutrona:Green:Hahn:"

Complete the function `ReadNames` below. If you don't understand how to use the input file stream, use *cin* instead for partial credit.

```

void ReadNames(istream & input, string & allnames, int & count)
// pre: input is bound to a file and open for reading, names in file are
//       one word each with no blanks in a name
// post: returns a string of all names separated by ":" and ending with ":",
//       and a count of the number of names read in
{

}

```

PART B (10 pts):

Write the function *FindKthName* whose header is given below. *FindKthName* is given a string of names in the format from the previous page (with a ":" after each name) and a number *k* and it returns the *k*th name in the string.

For example, if *names* had the value "Avila:Cutrona:Green:Hahn:", then *FindKthName*(3, *names*) would return **Green** and *FindKthName*(1, *names*) would return **Avila**.

The string class member functions are shown on page 2.

Complete the function *FindKthName* below.

```

string FindKthName(int k, const string & allnames)
// pre: kth name in allnames exists, ":" separates names and is the last
//       character in allnames
// post: returns kth name from allnames
{

}

```

PART C (10 pts):

Write the function *TwoNames* whose header is given below. *TwoNames* is given a string of names in the format from the previous page (with a ":" after each name) and the number of names in this string, and it returns by reference two unique randomly selected names from this string.

For example, if *names* had the value "Avila:Cutrona:Green:Hahn:", then *TwoNames*(*names*, 4, *n1*, *n2*) might return with *n1* having the value **Green** and *n2* having the value **Cutrona**, or *n1* having the value **Avila** and *n2* having the value **Green**, or any combination of two unique names. Note *n1* and *n2* could not have the same name.

The *Dice* class is shown on page 2.

In writing *TwoNames* you may call the function *FindKthName* from Part B. Assume *FindKthName* is correct, regardless of what you wrote in Part B.

Complete the function *TwoNames* below.

```

void TwoNames(const string & allnames, int number, string & name1,
              string & name2)
// pre: number is the number of names in allnames, number > 1,
//       the names in allnames are unique
// post: returns two different names from allnames randomly selected
{

```

```
}
```

PROBLEM 4 : (*The Olympics*: (12 pts))

A class *Olympics* is designed for being able to get information about the Olympics for a particular year. Each olympic event is assigned a unique number (starting at 1), and each athlete is assigned a unique number (also starting at 1).

Here is the definition of the *Olympic* class.

```
class Olympics
{
    public:
        Olympics(int year);           // constructor, info for year

        string NumberToEvent(int eventnumber) const;
        // post: given an event number, returns the name of the event

        string NumberToAthlete(int athletenumber) const;
        // post: given an athletenumber, returns athlete's name

        int Winner(int eventnumber, string medal) const;
        // pre: medal is gold, silver or bronze
        // post: returns athlete number of this medal for this eventnumber

        int NumEvents() const; // returns number of events
};
```

Below is an example usage of this class that prints the number of events for the year 2000 and the name of the athlete whose number is 10.

```
Olympics Sydney(2000);
cout << "number of events: " << Sydney.NumEvents() << endl;
cout << "Athlete 10 is " << Sydney.NumberToAthlete(10) << endl;
```

The output corresponding to this example is below.

```
number of events: 297
Athlete 10 is Marion Jones
```

Write the function *FindWinners* whose header is given below. *FindWinners* is given a year and the name of an event, and it prints the names of the gold, silver and bronze winners for this event.

For example, the call *FindWinners(2000, "women's 200m freestyle")* would output:

Susie O'Neal gold
Martina Moravcova silver
Claudia Poll bronze

Hint: You will need to search for the event number (events are numbered from 1 to the total number of events).

You can assume there are no ties, only one winner per medal per event.

Complete *FindWinners* below.

```
void FindWinners(int year, const string & event)
// pre: year is valid olympic year, event is valid event
// post: prints the names of the gold, silver and bronze winners for the event
//       in the specified year
{

}
```