

## Some Mandatory and Some Suggested Preparations

The introductory programming in C/C++ courses are taught from *Programming and Problem Solving with C++* by Dale, Weems and Headington. This text should be readily available at the academic bookstores in Blacksburg. The examination for credit will be based on the material covered in the first fifteen chapters of this book. Please note that there are many subtle issues of syntax and semantics in the C/C++ language. You are strongly advised to read this material carefully and to work through the questions provided at the end of each chapter.

The examination will be given in McBryde room 202 between 6:30 and 8:00 on Monday, August 24. In order to take the exam, you must:

- complete the programming assignment given below and submit your solution as specified at the end of this handout, before attempting the exam
- fill out the credit-by-exam form included with this handout and follow the instructions for advance payment of fees

In order to receive credit-by-exam you must achieve passing scores (70% or better) on **both** the programming assignment and the examination. **Note: averaging 70% or better will not guarantee credit.**

## Programming Project for Programming in C/C++: Multiple Choice Grading

Create a program to score results from a multiple-choice test, allowing partial credit for selected answers. The program will read the student ID numbers and their answers from the input file, ("anskey.dat"), followed by the test key. The program will then process the answers given by each student and determine two scores, one without partial credit and one with partial credit. The program will then produce a report file that includes the following information:

- ID number and both scores for each student.
- Averages of both sets of scores.
- A histogram showing how many students scored in each decile.

This information must be clearly labeled and formatted to be both readable and attractive.

## Input file description and sample:

The first line of the input file, ("anskey.dat"), contains two integers, separated by whitespace. The first integer specifies the number of students for whom data is given and the second specifies the number of questions given. Following the first line, the next <number of students> lines of the input file has the following format:

- A student ID number (9 contiguous digits).
- Two columns that should be disregarded.
- The responses provided by that student to the questions. Responses are expressed as single digits (0-9) with no intervening spaces. Note that a student may fail to respond to a question, indicated by a blank (space). It is not possible for a student to provide any response other than a single decimal digit or a blank.

Lines may have trailing whitespace (blanks and/or tabs) preceding the newline character. Each input line (except the first) will contain at least M characters before the newline, where  $M = (9 + 2 + \text{number of questions})$ . See the accompanying sample input file.

Following the last line of student data, there is an answer number header line that can be ignored. This line gives column headings 0-9 for the corresponding answer responses. The next <number of questions> key lines have the following format:

- Question number
- Character indicating the value of each possible response (10 in all). These characters will be separated by spaces and/or tabs. Possible characters and the corresponding value are as follows:

Character	Value
a	100%
b	70%
c	40%
d	20%
x	0%

There are some restrictions on the size of the input:

- There will be no more than 50 students.
- There will be no more than 50 questions.
- Each question will have at most 10 possible responses (0-9).

For instance:

20	20									
548932001	21255124001011000111									
229081368	21550166001011000111									
593072704	212544170020 1202111									
227131902	21250113001011000112									
230044180	21550163001011000111									
224235019	21251113002021202									
244577135	21550063001021000111									
368781746	21250013001011000111									
231135020	21210013001021202121									
219821682	21550163001011000111									
227253655	21250163001011000111									
426232843	21551164001011000141									
027158513	55550413001011000111									
006049924	21550163001011010111									
231454006	5510156001011101111									
743638994	21210013001021000111									
162747845	212544170020 1202111									
184376463	21250013001011000141									
273823564	51250413001011000212									
342541763	212551240010 1202111									
	0	1	2	3	4	5	6	7	8	9
1	x	x	a	x	x	x	x	x	x	x
2	c	a	c	x	x	x	x	x	x	x
3	x	x	x	x	x	a	x	x	x	x
4	x	x	x	x	x	a	x	x	x	x
5	a	c	x	x	x	x	x	x	x	x
6	x	a	x	x	x	x	x	x	x	x
7	x	b	b	x	x	x	a	x	x	x
8	x	x	c	a	a	c	x	x	x	a
9	a	x	c	c	x	x	x	x	x	x
10	a	x	x	a	x	x	x	x	a	x
11	x	a	b	c	d	x	x	x	x	x
12	a	x	x	x	x	x	x	x	x	x
13	x	a	x	x	x	x	x	x	x	x
14	x	a	x	x	x	x	x	x	x	x
15	a	x	b	x	x	x	x	x	x	x
16	a	x	x	x	x	x	x	x	x	x
17	a	x	x	x	x	x	x	x	x	x
18	x	a	x	a	x	x	x	x	x	x
19	x	a	x	x	x	x	x	x	x	x
20	x	a	x	x	x	x	x	x	x	x

### What to Calculate:

Each student response should be scored according to the table given above. Final scores, whether partial credit or not, should be reported as percentages, (out of 100), regardless of the number of questions asked. Scores and averages should be reported to two places after the decimal.

**Output description and sample:**

Your program **must** write all output data to a file named GRADER.OUT — use of any other output file name will again result in a massive deduction of points by the auto-grader. The sample output file shown below corresponds to the sample input data given above:

Dwight Barnette		
ID Number	Without Partial	With Partial
=====		
548932001	85.00	88.50
229081368	95.00	95.00
593072704	55.00	65.50
227131902	85.00	88.50
230044180	100.00	100.00
224235019	50.00	62.50
244577135	90.00	90.00
368781746	85.00	88.50
231135020	60.00	67.00
219821682	100.00	100.00
227253655	95.00	95.00
426232843	90.00	92.00
027158513	80.00	83.50
006049924	95.00	95.00
231454006	65.00	65.00
743638994	75.00	78.50
162747845	55.00	65.50
184376463	80.00	83.50
273823564	70.00	73.50
342541763	70.00	77.00
=====		
Average	79.00	82.70
Range	Count	
-----		
90:100	7	
80: 90	5	
70: 80	3	
60: 70	5	
50: 60	0	
40: 50	0	
30: 40	0	
20: 30	0	
10: 20	0	
0: 10	0	
-----		
# Students	20	

You are not required to use this exact horizontal spacing, but your output must satisfy the following requirements:

- The first line of the output file must contain **your** name.
- The second line of the output file must be blank.
- You **must** use the specified column labels student table and histogram table.
- All grades and averages **must** be formatted to show two places after the decimal.
- There must be two blank lines separating the student table and histogram data.
- The histogram values must be integers with no decimal places.
- The partial credit score must be used in determining the histogram decile counts.
- The ranges for the histogram should be treated as  $\geq 80 : < 90$ ,  $\geq 70 : < 80$ , etc., except for the first range which is  $\geq 90 : \leq 100$ .
- You **must** arrange your output in neatly aligned columns.
- You **must** use the same ordering of the columns as shown here.
- **Do not** insert any additional lines of output; **do** have a newline at the end of each line.

### Programming Standards:

You'll be expected to observe good programming/documentation standards. All the discussions in the text about formatting, structure, and commenting your code will be enforced. Specifically (but not comprehensively):

You must include a documentary header that specifies your name and ID number, the date the program was completed, and the specific compiler and operating system (e.g., MS C++ v5.0 under Windows NT). You may develop under different systems with different compilers, but all programs will be tested under MS Windows NT in Microsoft Visual C++ v5.0.

You must include a comment explaining the purpose of every variable you use in your program. Note: this applied to **all** functions, not just to main(). You must use meaningful, suggestive (of function!) variable names. Precede every major block of your code with a comment explaining its purpose. You don't have to describe **how** it works unless you do something so sneaky it deserves special recognition.

You must use indentation to make control structures like loops, if-else statements, and function bodies more readable.

Additionally, you are required to use at least five user-defined functions (besides main) in your program. Some suggestions:

- You must include header comments specifying your name, the compiler and operating system used and the date your source code and documentation were completed.
- The header comment must also include a brief description of the purpose of the program (sort of a user guide) — this should be in your own words, not copied from this specification.
- You must include a comment explaining the purpose of every variable you use in your program.
- You must use meaningful, suggestive (of function or purpose!) variable names.
- Precede every major block of your code with a comment explaining its purpose. You don't have to describe how it works unless you do something so sneaky it deserves special recognition.
- You must use indentation to make control structures, like loops, if-else statements, or function bodies more readable.
- Absolutely no file-scoped variables or functions are allowed! File-scoped constants and type declarations are acceptable.
- Use constants instead of variables where appropriate.
- You must pass parameters by reference only when the called function needs to modify the value of the parameter. Note this applies to array parameters as well (pass by constant reference).
- You must implement at least four functions.
- Not counting declarations and comments, the body of main() must contain no more than 20 lines.

Each function should be documented with a header that explains the purpose of each formal parameter, and include embedded comments as described above. You should use appropriate parameter-passing mechanisms at all times. Pass parameters by reference **only when necessary**.

You are also required to declare and use an enumerated type in your program, and to make use of a struct type to store the information you input and compute for each student.

### What to Turn In:

You are required to submit your solution in a 9x12 envelope, with your name and "C Credit-by-Exam" written on the outside. The envelope must contain the following:

- A printout of your C/C++ source code
- A printout of one input file and the corresponding output file that you used in testing your program. You may feel free to use the sample input above.
- A 3.5" diskette including your source code file(s) and at least one sample input file and corresponding output file you used to test your program.
- The disk must have a label that includes your name and ID number and the name of your source code file.
- A signed copy of the Honor Code Pledge given below.

### Honor Code Issues:

The examination on August 24 will be closed book and notes. You will be permitted only the use of pencil, paper and your mastery of the C/C++ language. In completing the programming assignment, you **may** consult any published C/C++ language references and/or programming guides you like. You **may not** consult other programmers, or program code produced by others, except for published examples. Consultation of koofers and/or programs written for C/C++ classes here or elsewhere is expressly forbidden. See the CS Department statement on koofers in the Survival Guide.

**VT Honor Pledge:** On my honor, I have neither given nor received unauthorized aid on this assignment.

---