University of Toronto at Mississauga
CSC 104S Spring 1998
Assignment 2
Turing Programming
Weight: 10%
Due: March 16, at the start of lecture

***** Erindale students must work in groups of 1 or 2 students *****

This assignment introduces you to computer programming using the Turing programming language. You will develop a working program that imitates the operation of an ATM (Automatic Teller Machine) of a bank. Since this is probably your first time writing a computer program, we will make it gentle by breaking your work into 5 progressive parts, with each part building upon the previous ones. For each part you complete successfully, you will receive some of the marks.

PART 1: Simple ATM

There is nothing better to build your confidence than writing a working program, so we will start with a program that can talk back to the customer and deliver the cash the customer requests.

Write a short Turing program that asks the customer how much cash the customer wants, and reports back to the customer the amount it has just sprewed out (use your imagination).

A sample dialog between your program (listed below in teletype font) and the customer (listed below in **bold** font) proceeds as follows:

```
***** Welcome to ATM *****

Please enter the amount of cash you want to withdraw: $80

Please pick up $80
```

Add comment lines at the top of your Turing program file with your full name and student number, course number (CSC 104), instructor name, TA name, and assignment part (PART 1). Save the program file using the name "a2part1.t" (The Save As command is in the File menu). Get a printout to hand in.

PART 2: Dream ATM

Now that your customer has had a good taste of your ATM, she would undoubtedly want more!

Modify the Turing program you wrote in PART 1 so that the ATM goes through a loop giving out as much cash as your customer wants, for as many times as your customer wants - it's the Dream ATM!

Every time through the loop, the program asks the customer how much cash he wants, and confirms the amount it has just sprewed out. It stops if the customer enters 0.

A sample dialog proceeds as follows:

***** Welcome to ATM *****

Please enter the amount of cash you want to withdraw: $**80**

Please pick up $80

Please enter the amount of cash you want to withdraw: $160

Please pick up $160

Please enter the amount of cash you want to withdraw: $0

Thank you for your business!

Note again that you should not write this Turing program from scratch, but rather should make a copy of a2part1.t, call it a2part2.t, and modify it to create the program for PART 2. Just like for PART 1, print a copy and hand it in.

PART 3: Limited ATM

Now you have to come back from dreaming to reality. Your ATM has a limited stock of cash, and can't just freely give out cash like paper napkins! Besides, your friendly bank wants to put a limit on even its best customer. In this part, you are to modify the Turing program for PART 2 so that it keeps track of how much cash your customer has obtained and quits if the next request would cause the total cash to exceed $200. So, the ATM-customer dialog proceeds as follows:

***** Welcome to ATM *****

Please enter the amount of cash you want to withdraw: $**80**

Please pick up $80

Please enter the amount of cash you want to withdraw: $160

Sorry - your request pushes you over your cash limit.
Thank you for your business!

Note that your program must also exit the with "Thank you" message when the customer enters a 0 amount. Print a copy of your program a2part3.t and hand it in.

PART 4: Considerate ATM

Modify your program for PART 3 so that the largest bills will be given to the customer. You can assume that the dollar amount is a multiple of $10. Only $100, $50, $20, and $10 bills will be given out by the ATM. Below is an example:

***** Welcome to ATM *****

Please enter the amount of cash you want to withdraw: $**40**

Please pick up $40: $20 $20

Please enter the amount of cash you want to withdraw: $130

Please pick up $130: $100 $20 $10

Please enter the amount of cash you want to withdraw: $50

Sorry - your request pushes you over your cash limit.
Thank you for your business!

Modify a2part3.t for this purpose, save it in file a2part4.t, and hand it in.

PART 5: Suspicious ATM

You'd only wish there were an ATM that would simply give you cash without checking who you are! In this part, you are to add some checking program statements by asking the customer to provide his/her account number and password typed in, and check whether this user is authorized.

A valid account number should contain nine digits. Its leading digit cannot be 0, and the sum of the nine digits should be divisible by 11. The password should simply be the account number with the nine digits in reverse order.

As a result, a sample dialog would have the following section just after " Welcome to ATM" and before your program asks for the dollar amount:

Please type in your account number: **187925264**
Please type in your password: 462529781

[Note: a *real ATM* wouldn't display the password you type in, but typically would display "**********". Since we are imitating ATM in software, you don't have to worry about this.]

If the account number is acceptable and the password is correct, then the ATM will proceed to ask for the amount; otherwise, it will give an error and quit. Below is an example dialog for the failure case:

Please type in your account number: **187725264**
Please type in your password: 462527781

Sorry your <account number, password> is not acceptable.

Modify your program for PART 4, a2part4.t, for this part, save your working program in a2part5.t, and hand in a copy.

Testing your Interactive Programs

When you are creating and testing your programs you will be typing the customer's parts of the dialog from the keyboard. In order for your TA to see that your program works properly, you must change your program so that it works properly using a file of test inputs. This will allow you to submit not only the program file but also the input files and the resulting output.

Once you have completed the highest numbered part which you can, use WinOOT to create a test file. This file should contain exactly the same input values that you previously typed on the keyboard when you were testing your program. Don't type any comments in the file. Save this file as "test1.dat" and use it with Run with Args inside the Run menu.

When you do this, you will notice that running a program on an input file is slightly different than using keyboard input. In Turing programs, characters entered from the keyboard are automatically echoed to the screen but characters entered from the data file are not. This means that once you are using an input file, the amounts your customer requests will not show up on the screen. You can fix this situation by adding one put statement in your program for every get statement. After you get a variable from the file, immediately put this variable to the output. Only make this change to the version of your program for which you are creating the test files.

When you are satisfied that your program is working properly on your input file, redirect the output (also from Run with Args) to a new filename. Call the output "test1.out". Produce a number of test files (test1.dat, test2.dat, etc.) with corresponding output files (test1.out, test2.out, etc.) that illustrate the various features of your program. Indicate on each file in pen or pencil what special case they are testing.

What to Hand In

Your submission should include the following:

- A header page based on the file "a2head" modified to include your own information.
- Printouts of a2part1.t, a2part2.t, a2part3.t, a2part4.t and a2part5.t
- A Printout of each test file followed by the corresponding output file. Don't forget the handwritten comments on the test files.

These pages should be stapled together.

Note that your programs should have some comments at difficult places to explain what they are doing. The comments should explain the programs *in English*, rather than paraphrasing the programs. Please be judicious in using comments: over comment is as bad as no comment at all.