CSI2131, Spring 2001
Assignment 2
Due on Friday, February 16, 11:00am

Worth: 7.5%   Marks out of 100

1 Written Problem 1: Compression using Huffman code 15 marks

1. Construct a Huffman tree for the following letter/frequency pairs:
   (e, 45), (h, 13), (i, 12), (l, 16), (o, 9), (p, 5).
   In case of a tie in frequencies, then devise your own tie-breaking strategy and stick to it. Label the edges of the tree so that each left edge receives the value "0" and each right edge receives the value "1". Use the convention that for any node in the tree, its left child is labeled with a smaller frequency than its right child.
2. Using the Huffman tree you built in Part 1, decode the following message: 101100
3. Using the Huffman tree you built in Part 1, encode the message "help".
4. Calculate the average number of bits per character when decoding a message containing the letters (with the corresponding frequencies) given in Part 1. Show your work.

2 Written Problem 2: Compression using Lempel-Ziv code 15 marks

1. Construct a Lempel-Ziv Encoding of the following string:
   "aabbcabcaaabcabcabcbaabcaa". Show the tree built from your encoding.
2. How many bits are necessary to encode the above message? How much space, then, gets saved by your encoding (in number of bits or bytes)?
3. Decode the following encoded message: "0a0b1b3b0c4b".

3 Programming Problem 1: De-compression using Huffman code 35 marks

A message has been posted for you in the website. In order to save disk-space and transmission time when downloading it, the message has been encoded using Huffman encoding and the file has been stored in binary format. You will be given a program containing what is necessary to build a Huffman tree based on the frequency of occurrence of each letter of the alphabet in English texts (for simplification, only capital letters and space characters are allowed in the original message). Your task is to write the part of the program that will allow you to decode the message based on this Huffman tree. You will complete the program that is provided in the web page. More details will be provided in file `README.txt`

In more detail your program will:

1. Read the binary file `encoded.bin` containing the encoded message.
2. Traverse the Huffman Tree (decoding the message) guided by the content of `encoded.bin`. **Note:** By convention, every left edge corresponds to 0 and every right edge corresponds to 1.
3. Write the decoded message in file `decoded.txt`

The program will be done by adding code to the program (project) provided in the web page. More information can be found in file `README.txt`

PROGRAMMING CONTEST: Prize and How to participate

The first student that can decode the given message using their Huffman decoding from the previous problem will win the following book as a prize:

`Lippman and Lajoie, ‘‘C++ Primer’’, Addison-Wesley, 1231 pages.`

Thanks to Addison-Wesley for donating this prize in support of the contest!

Rules for participation in the contest:

1. Check the web page to see if the news that somebody else already won the contest has been posted. In this case, ignore the next steps.
2. Send an e-mail to both:
   `nat@site.uottawa.ca` and `lucia@site.uottawa.ca` with your name and student number, the decoded message and all the files (as attachments) needed to run your program.
3. The winner will be the student whose e-mail is the one that first arrives at our mailbox, and that in addition conforms to the specifications of the previous 2 steps and contains a correct program and correct decoded message.

4 Programming Problem 2: Direct Addressing & Binary Search 35 marks

In this program, you are going to modify the sorted file generated in Assignment 1 (file with student numbers, passwords and names). Your program should iterate through a loop which will prompt the user for the following data:

```
student number:
old password:
new password:
confirm new password:
Continue or Exit the program (C/E):
```

For this assignment, we will pretend the input file is too big to fit into main memory. You must use binary search (the key is student number) to locate the appropriate record in which the password must be updated. In the binary search, you will examine each key by doing a direct access to the position in the file in which the corresponding record is stored; once the position is found, the record must be updated with the new password. Use methods `seekg` and `seekp` in order to move to the appropriate position in the file. A working solution of assignment#1 will be provided in the web page, which can be used as the basis for this program.