NYU, Tandon School of Engineering

Bridge to Computer Science Program

4th Exam

Thursday 16 December 2021

- You have two hours
- There are 100 points total.
- Note that there are longer problems at the end. Be sure to allow enough time for these.
- We supplied you with a file, named ‘solutions.txt’, where you should type all your answers.
- Write your name, netID and NYU ID at the head of the solutions file.
- For editing this file, you are allowed to use only plain text editors (Notepad for Windows users, or textEdit for Mac users).
- You are permitted to use Visual Studio (C++), CLion or XCode as compilers. And Textedit/Notepad for text editing
- Calculators are not allowed.
- This is a closed-book exam. No additional resourced are allowed.
- Pay special attention to the style of your code. Indent your code correctly, choose meaningful names for your variables, define constants where needed, choose most suitable control statements, etc.
- In all questions you may assume that the users enter inputs as they are asked. For example, if the program expects a positive integer, you may assume that users will enter positive integers.
- No need to document your code in this exam, but you may add comments if you think they are needed for clarity.
- Read every question completely before answering it.
- When done, please upload your answer file to Brightspace.nyu.edu, Gradescope and email to dkatz@nyu.edu

1) (3 pts) “new” is a function call in C++ which creates a new variable. What is its return value after creating the new variable?
   a. A Stack pointer
   b. The value that is created
   c. A virtual memory address
   d. A physical memory address

2) (3 pts) Which of the following memory allocation systems suffers from waste due to small amounts of memory in between allocations which cannot be used?
   a. Buddy System
   b. Fixed Partitioning
   c. Dynamic partitioning
   d. Paging

3) (3 pts) Which of the following OSI layers is, primarily, concerned with global addressing and routing?
   a. Application (7)
   b. Presentation (6)
   c. Transport (4)
   d. Network (3)

4) (3 pts)In order to identify an individual socket on the operating system, the transport layer has an address known as a ______________ in each packet which is a numeric value between 0 and 65535, there is one for the source and another for the destination.

5) (3 pts) ATM is a type of physical network which creates “permanent virtual connections” between endpoints. Packets (called cells in ATM) being sent over a PVC will always follow the same, predefined, path between start and end. This is known as a _____________ switched network.

6) (15 pts) TCP forms connections to allow reliable data transfer to occur. Explain what the sender and receiver should do if a single packet in the middle of the sliding window were lost in transit. (I.e. 10 packets were sent, each containing 1 byte of data and starting with sequence number 100, packet 5 is lost in transit but the rest arrive properly). Please use the above sequence numbers in your explaination.

7) (10 points) When one process calls “fork” the result is that two processes are created. List and describe the states that these two processes will be in at the conclusion of the fork call.

8) (15 pts) Generally speaking, the memory management requirement of “protection” prohibits sharing of pages in a paging system. However, for efficiency, we will often allow sharing. Explain a situation in which this might occur and how it could be implemented.

9) (10 pts) The hardware memory management unit in a paging system needs to do virtual to physical conversions frequently. If we decided to use a page size of 1,000 bytes rather than a, more common, 1024 this would slow the system significantly. Explain why it is important for performance to choose a page size which is a power of 2.

10) (15 pts) Deadlock prevention aims to avoid deadlocks by eliminating one of the four requirements for a deadlock. Explain how you would eliminate the "hold and wait" requirement in a system you were designing.

11) (20 pts) A hash table is a form of a data structure which aims to allow for (near) constant time insertions, searches and removals. It works by applying a "hash" function to every item that is inserted and determining a final position in a vector for that item and then placing it in that location in the vector. Obviously, this means that sufficient space is necessary for every item which will be inserted. An example of this might be a vector of size 10 where we insert the values 17,4,8 and 123. The hash function might simply mod the input value by the vector's size to find the final location of the item (i.e. 17%10 =7 so 17 belongs at position 7 in the vector). The result of the above insertions would be a vector which looks like:

			123	4			17	8	

Obviously this meets the requirements of constant time insertion, searching and removal. However, it suffers from a flaw in that multiple values can end up hashing to the same location (i.e. 17 and 107 and 77 and 37 will all hash to position 7), called a collision. To resolve this, we will use a list at each position in the vector. This may interfere with the constant time aspect but if we keep collisions to a minimum, performance shouldn't suffer.

Periodically, when we determine that performance is suffering (you will not be asked to detect performance issues) we will resize the vector. Unfortunately, that would necessitate creating a new, resized vector and re-inserting all of the values because they will, invariably, be at different locations once the vector's size changes. Resizing the above vector to size of 6, would result in the following:

		8	123	4	17

(i.e. 8%6=2, 123%6=3, %6=4 and 17%6=5)

Design a HashTable class which implements the above data structure for integers only. You should store the values in a vector of lists of ints (use the STL vector and list class). The default size of a HashTable should be 10 and the hash function, a private member, can be as described above (simple mod). You must implement the following functions:

- A default constructor
- insert – Takes one parameter, the integer to be inserted (do not worry about duplicates, we will never try to insert a duplicate value, you do not need to check for this)
- remove – Takes one parameter, the integer to be removed
- find – Takes one parameter, the integer to be searched, returns True or False
- hash – Takes one parameter, the integer to be hash, returns the integer "modded" by the vector size
- size – returns the number of values stored in the HashTable
- resize – takes one parameter, the new size, changes the size of the vector but maintains all of the inserted data in proper locations.