NYU, Tandon School of Engineering

Bridge to Computer Science Program

4th Exam

Thursday 15 December 2022

- 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++) 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) In a system that supports threads, which OS data structure would store a stack pointer?
   a. The PCB
   b. The TCB
   c. The PMT
   d. The heap
2) (3 pts) Which of the following function calls will cause the process to move from the running state to the ready state?
   a. sleep
   b. wait
   c. yield
   d. fopen

3) (3 pts) The cycle that the CPU performs for each instruction in a program is known as the:
   a. Ready – Running – Blocked cycle
   b. Fetch – Decode – Execute cycle
   c. Processor instruction set
   d. Hydrologic cycle

4) (3 pts) DNS provides a capability to resolve a name into an IP address and falls into which OSI layer?
   a. (this is not in any OSI layer)
   b. Application (7)
   c. Presentation (6)
   d. Transport (4)
   e. Network (3)

5) (3 pts) Which of the following is equivalent to the subnet mask 255.255.255.192
   a. /22
   b. /23
   c. /24
   d. /25
   e. /26

6) (3 pts) The clock memory management algorithm relies on ____________ to know what pages have been used recently?
   a. Use bits
   b. Present bits
   c. The Frame number (sequential order)
   d. The Urgent Pointer

7) (15 pts) Explain, in English, what is wrong with the following code and how you would fix it.

```
void printLine(int num) {
    for (int i = 0; i < num; i++)
        cout << '\t';
    cout << num;
}

void printManyLines(int num) {
    for (int i = 0; i < 1000; i++)
        printLine(num);
}
void createThread(int num) {
    //this function creates a kernel level thread
    //    and calls printManyLines passing it num
}

int main() {
    for (int i = 0; i < 5; i++)
        createThread(i);
}
```

8) (10 points) Explain how Classless InterDomain Routing (CIDR) overcomes the problem of wildly differing sizes of old "class" networks by explaining how you would design a network which was expected to have 350 hosts.

9) (10 pts) During an active TCP connection, we send a few packets (with increasing sequence numbers, of course) of data and then receive three ACKs for the same packet. Explain what has happened and explain what you would expect your TCP implementation to send in response.?

10) (10 pts) Explain how a DNS Iterative query is performed. In your answer, include the source and destination of each packet and what you expect to see in it.

11) (10 pts) In OS design a microkernel architecture is preferable to a monolithic kernel. Explain why this is true.

12) (10 pts) A hybrid approach to threading is sometimes used for systems that support both kernel and user level threads. Describe a situation in which you might use such a hybrid approach.

13) (17 pts) Given a Linked list similar to the one we designed in class, as shown below, we would like to overload the multiplication operator to accept a positive integer and will return a list which is the list repeated multiple times. For example, if the original list is [1,2,3], multiplied by 3 would be [1,2,3,1,2,3,1,2,3]

```
template <class T>
class LList;
template <class T>
class LListNode {
    T data;
    LListNode<T>* next;
public:
    LListNode(const T& item = T(), LListNode<T>* newnext = null) {
        data = item;
        next = newnext;
    }
    friend class LList < T >;
};
template <class T>
class LList {
    LListNode<T>* head;
    LListNode<T>* recursiveCopy(LListNode<T>* rhsHead);
public:
    LList() :head(nullptr) {}
    ~LList() { clear(); }
    LList(const LList<T>& rhs) :head(nullptr) { *this = rhs; }
    LList& operator=(const LList& rhs);
    void insertAtHead(const T& item);
    T removeFromHead();
    bool isEmpty() { return head == nullptr; }
    void clear() { while (!isEmpty()) removeFromHead(); }
    int size()const;
};
```