Systemnahes Programmieren
Prüfungsvorbereitung

Teil 1: C# Grundlagen
Folgende Informationen stammen aus Interviews mit Studenten, welche die PRGSY Prüfung bereits absolviert haben. Über den exakten Umfang der Prüfung kann deshalb nur gerätselt werden.

Speicherstrukturen: Funktionsweise von Heap und Stack. Unterschied ValueType und ReferenceType. Wie Klassen, Strukturen und Datentypen im Memory abgelegt. Womöglich sind auch Skizzen verlangt.
Siehe dazu auch folgenden Artikel: C# Corner - Memory Management in .NET.

Algorithmen: Soll scheinbar eine ziemlich umfangreiche Aufgabe kommen. Beispiel: Ein Bitmap um 90° drehen ohne dabei auf bestehende Graphics Klassen zurückzugreifen. Irgendein beschriebenes mathematisches Problem lösen. Bit-Shifting wird höchstwahrscheinlich auch ein Thema sein. Beispiel: Methode entwickeln, welche einen bestimmten Bereich eines uint Bitstrings auf 0 bzw. 1 setzt. Oder: Einen uint Bitstring um eine vorgegebene Anzahl stellen schieben. Den positiven oder negativen Überlauf jeweils wieder am anderen Ende ansetzen.

Robotermethode: Mit vorgegebenen Methoden, welche im Verlauf des Moduls bereits im Zusammenhang mit dem C# Roboter verwendet wurden, eine bestimmte Aufgabe lösen. Zum Einsatz kommen sicher die Trigometrie-Formeln (sin, cos, usw..). Methoden für die Bewältigung von Fahrbefehlen usw.. könnten verlangt werden.

Teil 2: Parallele und verteilte Programmierung

Aufgabe 1: Kreuzworträtsel / Multiple Choice
3. Die explizite Synchronisation ist implizit…
ATOMIC

5. Ein nebenläufiges Programm ist korrekt, wenn es….ist.
SICHER

7. Sprachkonstrukt für die explizite Synchronisation:…
AWAIT

8. Sprachkonstrukt für nebenläufige Ausführung.
CON

1. Sicherheit in einem parallelen Programm heisst kein…
DEADLOCK

2. C# Symbol, das nur einen Thread in einen kritischen Abschnitt zulässt.
LOCK

3. Sprachkonstrukt für atomare Aktion
ATOMAR

4. Ein nebenläufiges Programm ist korrekt, wenn es …. Ist.
LEBENDIG

5. Lebendigkeit in einem parallelen Programm heisst keine…hat
STARVATION

6. C# Anweisung, die nur einen Thread oder Prozess in einem kritischen Abschnitt zulässt.
MUTEX

Aufgabe 2: Semaphore
Semaphore Klasse programmiere mit limiterter Anzahl Threads. Siehe Aufgabe. Stichworte: Release, Aquire.
Aufgabe: Ordnen Sie, falls möglich, die folgenden Aussagen den Zeilennummern im Code zu:

a. Der Thread befindet sich im newZustand.
→ Auf der Zeile der Instanzierung mit "new".

b. Der Thread wird in den ReadyZustand versetzt.
→ Nach der Instanzierung mit "new".

c. Diese Operation(en) des Threads werden (quasi)parallel abgearbeitet.
→ Alles war in der mit dem Thread instanzierten Methode ausgeführt wird (z.B. "Run()" oder "Go()" Methode).

d. Der Thread wird in jedem Fall in den "blocked" Zustand versetzt.
→ Bei sleep, join und I/O wird ein Thread in den "blocked" Zustand versetzt. In der Prüfungsaufgabe kann es auch sein, dass er nie in den blocked Zustand kommt.

e. Der Thread kommt vom "Running" Zustand in den ObjectsWaitPool.
→ Monitor.Wait(this)

f. Der Thread kommt vom "Running" Zustand in den ObjectsLockPool.
→ lock(this) in der Methode "Aquire".

g. Der Thread kommt vom ObjectsWaitPool in den ObjectsLockPool.
→ Monitor.Pulse(this).

h. Der Thread wird abgebrochen.
→ thread.Abort();

i. Der Thread setzt ein Timeout.
→ Theoretisch bei Monitor.Wait(…) möglich ein Timeout zu setzen.
→ An der Prüfung auch möglich, dass er nirgends eines setzt.

k. Der Thread wird in den DeadZustand versetzt.
→ thread.Abort()

l. Der Thread gibt freiwillig CPU frei und wechselt in den Ready Zustand.
→ thread.Yield() oder thread.Sleep(0)

Aufgabe 3
Ausgangslage: Ein Thread befindet sich in Monitor.Wait(this). Dabei wird er mit thread.Abort() aus diesem Zustand gerissen. Trotzdem wird später ausgegeben, dass noch 1 Thread im Running Zustand sei.
→ Hier geht es darum, dass Codezeilen nach dem Monitor.Wait(..:) nicht mehr ausgeführt werden. Abilfe schafft das Konstrukt try{}finally{}.

Aufgabe 4
Schreiben Sie die Klasse BoundedStack so, dass sie threadsicher und prozessübergreifende Synchronisation implementiert.
→ Semaphore und Mutex lassen prozessübergreifende Synchronisation zu.
→ Für einen boundedBuffer verwenden wir zwei Semaphore (pop und push) mit einem systemeindeutigen String:

```
semaPush = new Semaphore(size, size, "uniquestring");
semaPop = new Semaphore(0, size, "uniquestring");
```

→ WaitOne zum Warten bis Semaphore frei ist.
→ Beim Lesen/Schreiben auf Stack muss gelockt werden: lock(this)

Aufagbe 5: Echo Server
A. Welche Einschränkungen besitzt die Klasse EchoServer ? Geben Sie nur zwei Einschränkungen an.
→ Keine Exceptions implementiert, stürzt bei kleinen Fehlern schon ab, völlig instabil
→ TcpListener listen = new TcpListener(IPAddress.Loopback, port) erlaubt nur lokale Anfragen.
→ Server kann (je nach je) nur eine einzige Anfrage beantworten.
→ Server kann nur TcpStreams verbinden.

B. Mit welchem Konstrukt könnte eine der Einschränkungen der SimpleCheckDate behoben werden?
→ Mit Threads könnte der Server mehrere Anfragen gleichzeitig bearbeitet werden.
→ ThreadPool um den Server skalierbarer zu machen.
→ Exception Handling mit try/catch.

C. Meldung “Port already in use” erscheint.
→ Ein Port darf nicht doppelt belegt werden.

D. Welches Kommando wird verwendet, um via telnet auf den Server zuzugreiffen:
telnet servername 4711
oder
telnet
> o servername 4711

Danach können Datenstrings gesendet werden, welche vom Server bearbeitet werden.
E. Wenn der Client ein Web Browser ist, wie würde die Anfrage aussehen?
→ GET HTTP/1.1
→ Der Server kann (je nach je) nicht damit umgehen und gibt einen Fehler zurück.

F. Wie würde ein Client aussehen, der diese Aufgabe lösen könnte?
→ TcpClient client = new TcpClient(servername, port);

Aufgabe 6: Daytime Server
Gegeben ist der Code eines HTTP Servers.

A. Anfrage des Browsers läuft ins Leere. Warum?
→ Server läuft am falschen Port oder Client fragt am falschen Port. Siehe AbstractServer.
→ Wird kein Flush gemacht, schreibt ein StreamWriter keine Meldung zurück.
sw.WriteLine(msg);
sw.Flush();

B. Bei ReadRequest wird die Anfrage abgebrochen. Warum?
→ Möglichweise hat der Client die Verbindung geschlossen, vor die Anfrage beantwortet werden konnte.

C. Was muss gemacht werden, damit nur eine bestimmte URL akzeptiert wird?
→ In ReadRequest auf Vorkommnis eines bestimmten Zeichnes/URL prüfen.

D. Der Server ist ein Dienst auf einem entfernten Server. Wie stellen Sie sicher, dass dieser Server auch bei vielen Anfragen nicht zu viel Resourcen frisst?
→ Mit ThreadPool, um den Server nicht zu überlasten.
→ Buffergrösse des ThreadPools so gross wie möglich, damit keine Anfragen fallen gelassen werden müssen.
→ So wenig Threads wie möglich kreieren, damit nicht viel CPU Zeit beim Taskwechsel verloren geht.