If you are asked to *briefly* describe or answer something, the answer can reasonably be provided in about two sentences. When completing the program fragments, write your code on the normal paper, not on the paper with the program text.

Task 1 (6):

(i) State Amdahl's Law by giving the formula for $S_p(n)$, and briefly explain its components.

(ii) What is the definition of efficiency, $E_p(n)$? Simply give the formula.

(iii) Briefly explain: Why is usually $E_p(n) < 1$? How is it possible that, in rare cases, $E_p(n) > 1$?

Task 2 (4):

(a) Draw a 2-dimensional CCC network, giving the nodes their canonical labels.

(b) *Choice:* Choice!

(1) Formally define the k-dimensional Omega network.

(2) Formally define the k-dimensional Butterfly network.

Task 3 (5): Briefly answer the following questions.

(i) What is the difference between blocking and non-blocking receiving in MPI to the programmer, besides the names of the functions?

(ii) What is the difference between non-blocking buffered send and non-blocking synchronous send in MPI to the programmer, besides the names of the functions?

(iii) What is the practical relevance of bisection bandwidth in a network?

(iv) What is the difference between a cluster and a constellation?

(v) A *ready send* offers the MPI implementation on the sender's part additional knowledge about the receiver. Why can this knowledge be advantageous?

Task 4 (5): *Choice:* Choice!

(1) Coffman described three necessary conditions on a system such that a deadlock situation potentially can occur. Name and briefly describe each condition.

(2) Give the proof that XY-Routing on an acyclic grid is deadlockfree. You do not need to precisely define the labeling function.

Task 5 (2): Let the alignment and the width of `double`, `int`, `float`, `short` be $8, 4, 4, 2$, respectively. Given the following definition, what is the type map of `v`, starting with offset 0?

```
typedef struct { int a; short b; float c[2]; } one_t;
typedef struct { one_t d; double e[2]; } two_t;
two_t v;
```

Task 6 (8): *Choice:* Choice!

(1) Describe the BSP model with its defining parameters.

(2) Describe the LogP model with its name-giving parameters.

Task 7 (5): Let the variable `data` point to an array consisting of key/value `int`-triples:

```
struct { int key; int value[2]; } *data;
```

The size of the array shall be `n`. Write C-code that creates an `MPI::Datatype values` and a `void*p`, such that `Send(`*p*`,1,`*values*`,dest,tag)` sends exactly all `value[0]` items of the array in the sequence of their locations. Do not copy or rearrange any data.

Task 8 (10): Consider the MPI program on the attached sheet. You are required to fill in the bodies of the missing functions, the semantics of which are explained in their comments.

- All processes know that they shall perform that communication, and all enter the correct function.
- Upon entry of the function, a process knows only that what it is passed in the function's parameters and what is in the global variables printed on the sheet. The global variables are already initialised. If you need datatypes or reduction functions, you have to create and free them for each function.
- Each function shall be implemented with at most two dynamic calls to an MPI communication function per process (including all calls from any helper functions).

Task 9 (13): Write a threaded program that implements a set of five worker threads, executing functions which are specified externally, with a master thread. Do so by completing the indicated functions of the fragment of the threaded program on the attached sheet.

- The worker threads have been created in a main program, which afterwards transfered control to `master_thread`. Only these threads access the global data.
- If you need synchronisation structures, add them in global variable space. You may assume that these structures are initialised with default parameters and freed correctly before/after your functions are called; if you need special initialisation, indicate this.
- For simplicity, we assume the threads to run forever and not be cancelled.
- Skip the *pthread_* prefixes on Pthread names.

Task 10 (13): Suppose a single broadcast shall be performed on a graph of n nodes, where each node can send on *only one channel* per time step. How many time steps are needed at minimum to complete the broadcast, ...

(a) ... if the graph is complete?

(b) ... if the graph is a complete binary tree, and the tree's root is the broadcast's root?

(c) ... if the graph is a complete binary tree, and the broadcast's root is such that the minimal time is maximal?

Proove your claims.

Task 11 (4): Briefly answer the following questions.

(i) What is the difference between user-level threads and kernel-level threads?

(ii) What are advantages of each of the two thread types (one advantage each)?

Wenn Sie etwas *kurz* erklären oder beantworten sollen, so können etwa zwei Sätze dafür ausreichen. Für die Aufgaben, bei denen Sie Programmcode vervollständigen sollen, schreiben Sie Ihre Lösung auf das normale Klausurpapier (nicht auf das Blatt mit dem Programmtext).

Aufgabe 1 (6):

(i) Geben Sie Amdahls Gesetz an, indem Sie die Formel für $S_p(n)$ angeben und kurz ihre Teile erklären.

(ii) Geben Sie die Formel zur Definition der Effizienz $E_p(n)$ an.

(iii) Erklären Sie kurz: Warum ist normalerweise $E_p(n) < 1$? Warum ist in manchen Fällen $E_p(n) > 1$?

Aufgabe 2 (4):

(a) Zeichnen Sie ein 2-dimensionales CCC-Netzwerk. Geben Sie dabei den Knoten die kanonischen Bezeichnungen.

(b) *Auswahl:* Auswahl!

(1) Geben Sie die formale Definition des k-dimensionalen Omega-Netzwerkes an.

(2) Geben Sie die formale Definition des k-dimensionalen Butterfly-Netzwerkes an.

Aufgabe 3 (5): Beantworten Sie kurz die folgenden Fragen.

(i) Was ist der Unterschied für den Programmierer zwischen blocking und non-blocking receive in MPI, abgesehen von den Funktionsnamen?

(ii) Was ist der Unterschied für den Programmierer zwischen non-blocking buffered send und non-blocking synchronous send in MPI, abgesehen von den Funktionsnamen?

(iii) Was ist die praktische Bedeutung der Bisektionsbandbreite eines Netzes?

(iv) Was ist der Unterschied zwischen einem Cluster und einer Constellation?

(v) Die MPI-Implementierung auf Seite des Senders erhält zusätzliche Informationen über den Empfänger, wenn ein *ready send* verwendet wird. Wie kann diese Information nützlich sein?

Aufgabe 4 (5): *Auswahl:* Auswahl!

(1) Nennen und beschreiben Sie kurz die drei notwendigen Bedingungen nach Coffman für die Möglichkeit eines Deadlocks.

(2) Geben Sie den Beweis der Deadlockfreiheit beim XY-Routing an. Sie müssen die Bewertungsfunktion dabei nicht präzise angeben.

Aufgabe 5 (2): Seien Alignment und Größe der Datentypen `double`, `int`, `float` und `short` 8, 4, 4 bzw. 2. Was ist die type map von `v`, beginnend mit Offset 0?

```
typedef struct { int a; short b; float c[2]; } one_t;
typedef struct { one_t d; double e[2]; } two_t;
two_t v;
```

Aufgabe 6 (8): *Auswahl:* Auswahl!

(1) Beschreiben Sie das BSP-Modell und dessen definierende Parameter.

(2) Beschreiben Sie das LogP-Modell und dessen namensgebende Parameter.

Aufgabe 7 (5): Die Variable `data` zeigt auf ein Feld aus Schlüssel/Wert `int`-Tripeln:

```
struct { int key; int value[2]; } *data;
```

Die Grösse dieses Feldes sei `n`. Schreiben Sie C-code, der `MPI::Datatype values` und `void*p` erzeugt, sodaß `Send(`*p*`,1,`*values*`,dest,tag)` genau alle `value[0]`-Elemente des Feldes in Reihenfolge ihrer Ablage im Speicher sendet. Sie dürfen die Daten nicht kopieren oder umordnen.

Aufgabe 8 (10): Fügen Sie die Funktionsrümpfe zu den Funktionen auf dem MPI-Programm auf dem beigelegten Blatt hinzu. Die Semantik der Funktionen ist in den jeweiligen Kommentaren spezifiziert.

- Alle Prozesse wissen, welche Kommunikation durchgeführt werden soll, und alle rufen die jeweilige Funktion auf.
- Beim Funktionseintritt weiß ein Prozeß nur das, was die Funktion als Parameter erhält und was in den angegebenen globalen Variablen steht. Alle globalen Variablen sind wie angegeben initialisiert. Wenn Sie spezielle Datentypen oder Rekursionsfunktionen benötigen, müssen Sie sie jeweils neu erzeugen und freigeben.
- Für die Implementierung einer Funktion dürfen Sie höchstens zwei dynamische Aufrufe von MPI-Kommunikationsfunktionen pro Prozeß verwenden. Dies schließt alle Aufrufe in eventuell von Ihnen definierten Hilfsfunktionen ein.

Aufgabe 9 (13): Schreiben Sie ein Programm, welches eine Menge von Arbeiter-Threads implementiert, an die extern spezifizierte Funktionen von einem Master-Thread zur Ausführung übergeben werden. Vervollständigen Sie dazu das Fragment des Programmes auf dem beiliegenden Blatt an den angegebenen Stellen.

- Die Arbeiter-Threads wurden im Hauptprogramm erzeugt. Dieses rief anschliessend die Funktion `master_thread` auf. Andere Threads greifen nicht auf die globalen Daten zu.
- Wenn Sie Variablen zur Synchronisation benötigen, fügen Sie diese zu den globalen Variablen hinzu. Sie dürfen voraussetzen, daß diese Variablen bevor/nachdem Ihre Funktionen aufgerufen werden, standardmäßig initialisiert bzw. gelöscht werden; wenn Sie eine besondere Initialisierung benötigen, geben Sie dies an.
- Zur Vereinfachung gehen wir davon aus, daß die Threads nicht abgebrochen werden und also niemals enden.
- Lassen Sie die *pthread_*-Präfixe der Pthread-Namen weg.

Aufgabe 10 (13): Ein Einzelbroadcast werde auf einem Graphen mit n Knoten implementiert. Jeder Knoten kann in jedem Zeitschritt auf *höchstens einem Kanal* senden. Wieviele Schritte benötigt man mindestens für den Broadcast, wenn …

(a) … der Graph vollständig ist?

(b) … der Graph ein vollständiger Binärbaum ist, und der Broadcast von der Wurzel des Graphen ausgeht?

(c) … der Graph ein vollständiger Binärbaum ist, und der Broadcast von einem Knoten ausgeht, bei dem die minimale Zeit maximal ist?

Beweisen Sie Ihre Behauptungen.

Aufgabe 11 (4): Beantworten Sie kurz die folgenden Fragen.

(i) Was ist der Unterschied zwischen User-Level Threads und Kernel-Level Threads?

(ii) Was sind die Vorteile dieser beiden Spielarten (einer pro Art)?