**Множество реализаций алгоритма, заданного последовательной программой**, …

1. Пустое
2. Состоит из конечного числа элементов, но в вопросе недостаточно информации, чтобы указать количество
3. Счётное
4. **Состоит из 1 элемента**

Пояснение: реализация – это конкретный вариант исполнения последовательности операций, не противоречащей частичному порядку, а поскольку программа последовательная, то реализация может быть только одна.

Вопрос 2

**Взаимное исключение – это:**

1. **Порядок работы параллельных процессов с некоторым ресурсом, разрешающий допуск к ресурсу в один момент только одному процессу из множества процессов.**
2. Ситуация исключения доступа, в которой ни один из процессов не может получить доступ к ресурсу, если процессы обратились к ресурсу одновременно.
3. Ситуация, в которой при требовании одним из процессов параллельной программы исключить некоторый другой процесс из множества процессов, другой процесс поступает взаимно и требует исключения первого.

Вопрос 3

Для того чтобы каждый из двух процессов имел гарантированный шанс на доступ к некоторому неразделяемому ресурсу, достаточно, чтобы существовали операции

* **try\_and\_lock(r)** – захватывает ресурс r, если он в настоящее время не захвачен, либо возвращает управление без доступа к ресурсу;
* **unlock(r)** – освобождает ранее захваченный ресурс r.

1. Верно
2. **Неверно**

Пояснение: есть ситуация, что один поток может никогда не успевать захватить ресурс. Это называется голоданием (Starvation). Тут нужен алгоритм предотвращения голодания, например, очередь из процессов.

Вопрос 4

**Чем отличается алгоритм от программы?**

1. **Программа – это представление алгоритма на входном языке вычислительной системы (исполнителя).**
2. **Для алгоритма существует счётное множество программ, которые задают его реализацию на исполнителях.**
3. В программе задано управление на множестве операций алгоритма, а в алгоритме управления нет.

Пояснение к 3: в алгоритме есть частичный порядок, который говорит, как правильно исполнять программы.

Вопрос 5

**Определение дедлока:**

1. Недостаток ресурсов для исполнения программы.
2. Ошибка в управлении, вызвавшая недостаток ресурсов.
3. **Ситуация, в которой несколько параллельных процессов не могут продолжать выполнение, так как каждому из них для продолжения требуется доступ к ресурсам, который ограничили другие процессы.**

Вопрос 6

Динамическая балансировка **LuNA** **{учитывает соседство данных и вычислений,** информационные зависимости между фрагментами вычислений, функции, вычисляемые фрагментами вычислений **}** за счёт **{отображения фрагментов на виртуальный вычислитель**, отсутствия побочных эффектов у фрагментов вычислений, особенностей сетевой топологии, применения Гильбертовой прямой **}** и **{является масштабируемой}**.

*/\*асинхронно, не централизовано, автоматичесокой декомпозиции данных нет\*/*

Вопрос 7

**Управление** – это **{бинарное}** отношение **{частичного}** порядка на множестве F-операций алгоритма A.

Вопрос 8

**Разделяемый ресурс** – это

1. Ресурс, который можно разделить на части и потреблять постепенно.
2. Ресурс, для возможности совместного использования которого процессы должны использовать примитивы взаимного исключения.
3. Ресурс, доступ к которому процессы получают только поочерёдно на основе принципа разделения времени.
4. **Ресурс, который несколько процессов используют совместно, при этом в зависимости от ресурса и характера операций может требоваться взаимное исключение доступа.**

Разделяемый ресурс – это

1. Ресурс, доступ к которому процессы получают только поочерёдно на основе принципа разделения времени.
2. Ресурс, который можно разделить на части и потреблять постепенно.
3. Ресурс, для возможности совместного использования которого процессы должны использовать примитивы взаимного исключения.
4. **Ресурс, который одновременно могут использовать несколько процессов.**

Пояснение к 1: разделяемый – shared, разделить в контексте пункта – split.

Вопрос 9

**Технология фрагментированного программирования** предназначена для автоматического конструирования **{программы}** для **{суперкомпьютеров}** из описания **{алгоритма}** в области задач **{численного моделирования}**.

Вопрос 10

**Сколько различных алгоритмов** можно построить для покомпонентного сложения двух одномерных векторов размера n X([1], [2],...,[n])

1. Конечное количество.
2. Потенциально бесконечное количество.
3. **Счётное количество.**

Пояснение: несмотря на то что количество вариантов сложения конечно, мы всегда можем модифицировать алгоритм добавлением сложения и последующего вычета какого-нибудь числа. Алгоритм останется правильным, несмотря даже на наличие такой бесполезной операции.

Вопрос 11

**Выберите правильные высказывания:**

1. **Частично рекурсивная функция – функция, построенная из простейших вычислимых функций конечным числом применений операторов суперпозиции, примитивной рекурсии и минимизации.**
2. **Примитивно рекурсивные функции всюду определены.**
3. Частично рекурсивная функция – простейшая функция, построенная из операторов суперпозиции, примитивной рекурсии и минимизации.
4. Примитивно рекурсивная функция – функция, построенная из простейших функций конечным числом применений операторов суперпозиции и минимизации.
5. **Примитивно рекурсивная функция – функция, построенная конечным числом применений операторов суперпозиции и примитивной рекурсии.**
6. Любой алгоритм задаёт способ вычисления некоторой примитивно рекурсивной функции.

Пояснение: просто вспоминаем матлог)

Вопрос 12

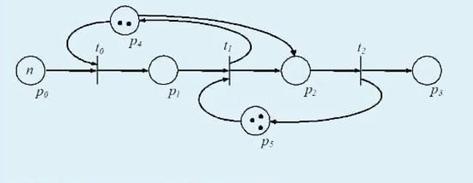
**Фрагментированный алгоритм** – это **{конечный или потенциально бесконечный}** */\*с не фрагментированным также\*/* **{ориентированный}** двудольный граф, вершинами которого являются фрагменты данных и вычислений.

Вопрос 13

**Фрагментированный алгоритм** – это форма представления алгоритма, ориентированная на

1. Автоматическое распараллеливание последовательного кода
2. Автоматическую декомпозицию данных и вычислений
3. **Автоматизацию обеспечения нефункциональных свойств генерируемых параллельных программ.**

Вопрос 14

На рисунке задано конвейерное управление.

1. Верно
2. **Неверно**

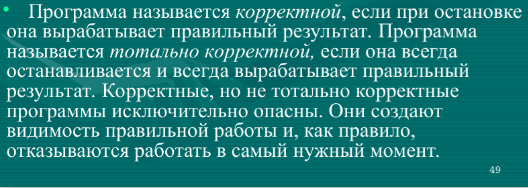
Пояснение: дуга (p4,p2) рушит двудольность графа. Эти места забыли разделить переходом.

Вопрос 15

**Корректной программой** называется программа, которая определяет множество реализаций алгоритма, **каждая из которых** (реализаций) приводит к завершению вычисления и вырабатывает при этом правильный результат.

1. Верно
2. **Неверно** */\*это тотально корректная\*/*

Серьёзное замечание: на записи видно, что система забраковала ответ “Верно”. Городничев как-то сам запутался, верно ли это определение. Возможно, что сверху определение тотально корректной программы, но Городничев решил, что ответ должен быть “Верно”. Однако в презентациях чётко написано, что это относится к тотально корректной программе:



Вопрос 16

**Функциональные термы** – **{слова}** особого вида, записанные в функциональном **{алфавите}**.

Вопрос 17

**В технологии фрагментированного программирования** пользователь освобождается от:

1. Рутинной работы параллельного программирования
2. Всех задач параллельного программирования
3. **Некоторых задач параллельного программирования, как рутинных, так и сложных.**

Вопрос 18

**Для доказательства корректности параллельной** программы следует показать, что на специально подготовленных различных наборах входных данных при всех осуществленных запусках программы она выдала корректный результат.

1. Верно
2. **Неверно**

Пояснение: может быть такое, что на одной машине мы можем не попадать в условия, при которых будет получена ошибка. Более того, тестированием невозможно доказать корректность параллельной программы из-за недетерминизма.

Вопрос 19

**Граф достижимости Петри** определяет все достижимые **{разметки}** и **{последовательности срабатываний переходов}**, приводящие к ним.

*/\*разметка - строка, описывающая распределение фишек в распределении Петри\*/*

Вопрос 20

Отметьте необходимые **условия для возникновения дедлока** в системе параллельных взаимодействующих процессов:

1. **Дозахват ресурса: для продолжения выполнения процесса требуется захват дополнительного ресурса.**
2. **Циклическое ожидание: если имеется 2 процесса, то каждый из них ждёт освобождения ресурса другим процессом; если более, то имеются циклы вида А ждёт Б, Б ждёт В, В ждёт А и т. П.**
3. **Взаимное исключение: ресурсы, которые требуются процессам для продолжения вычислений, должны использоваться процессами одновременно.**
4. Наличие недостижимых вершин в графе достижимости сети Петри: такие вершины соответствуют состоянию дедлока.
5. **Не допускается временное освобождение уже захваченных ресурсов: процессы не готовы освободить уже захваченные ресурсы.**
6. В программе MPI используются только блокирующие функции для обмена сообщениями.
7. Применяются функции интерфейсов POSIX Threads для блокировки ресурсов.

**////////////////////////////////////////////////////////////////////////////////////**

Вопрос 1

**Частично корректной программой** называется программа, которая определяет множество реализаций алгоритма, каждая из которых (реализаций) приводит к завершению вычисления и вырабатывает при этом правильный результат.

1. True
2. **False**

*/\*для любых нач. данных, удовлетворяющих определённому условию, результат вычисления (если оно завершается), удовлетворяет опр. условию\*/*

Вопрос 2

**P – «часть» вычислений**, заданных алгоритмом, которую можно выполнить параллельно, (1–P) – «часть» последовательных операций, S[1][N] – ускорение, которое может быть достигнуто при выполнении параллельной программы на N процессорных элементах по сравнению с производительностью программы, работающей на 1 процессорном элементе. Правильная формула для **закона Амдала**:

1. **S[1][N] = 1 / ((1 - P) + P / N)**
2. S[1][N] = P / ((1 - P) + P / N)
3. S[1][N] = 1 / (1 + P / N)

Вопрос 3

В чём **преимущество неблокирующих функций** в MPI перед блокирующими?

1. **Их можно использовать для осуществления связи на фоне выполнения вычислительных операций, то есть для наложения во времени коммуникаций и вычислений**.
2. Они используют специальный протокол для более быстрой отправки байтов.
3. Использование этих команд делает текст программы короче, чем при блокировке функций отправки/получения.

Вопрос 4

Возможные результаты работы **программы**

#include <omp.h>

#include <iostream>

int main() {

  int a = 123;

#pragma omp parallel num\_threads(2)

  {

    int thread\_id = omp\_get\_thread\_num();

    int b = (thread\_id + 1)\*10;

    a += b;

  }

  std::cout << “a = “ << a << “\n”;

  return 0;

 }

1. a = 123
2. **a = 133** *//проработает 0-й поток*
3. **a = 143** *//проработает 1-й поток*
4. **a = 153** *//проработают оба потока*
5. a = 0

Вопрос 5

**Модель OpenMP** определяет, как потоки параллельной программы взаимодействуют с помощью **{общей памяти}**.

Вопрос 6

**Модель MPI** определяет, что процессы параллельной программы используют **{сообщения}** для взаимодействия между собой.