**Assignment # 3**

**Question # 1**

What resources are used when a thread is created? How do they differ from those used when a process is created?

**Question # 2**

Using Amdahl’s Law, calculate the speedup gain of an application that has a 60 percent parallel component for (a) two processing cores and (b) four processing cores (c) eight processing cores.

**Question # 3**

Which of the following components of program state are shared across threads in a multithreaded process? a. Register values b. Heap memory c. Global variables d. Stack memory

**Question # 4**

|  |
| --- |
| Run this code in Ubuntu and check output. |
| #include <iostream>  #include <pthread.h>  #include <vector>  using namespace std;  // Function to calculate factorial  void\* factorial(void\* arg) {  int num = \*(int\*)arg;  long long result = 1;  for (int i = 1; i <= num; ++i) {  result \*= i;  }  cout << "Factorial of " << num << " is " << result << endl;  return nullptr;  }  // Function to calculate Fibonacci series  void\* fibonacci(void\* arg) {  int n = \*(int\*)arg;  vector<int> fib(n);  fib[0] = 0;  if (n > 1) {  fib[1] = 1;  }  for (int i = 2; i < n; ++i) {  fib[i] = fib[i - 1] + fib[i - 2];  }  cout << "Fibonacci series up to " << n << ": ";  for (int i = 0; i < n; ++i) {  cout << fib[i] << " ";  }  cout << endl;  return nullptr;  }  // Function to sum two numbers  void\* sum(void\* arg) {  int\* nums = (int\*)arg;  int sum\_result = nums[0] + nums[1];  cout << "Sum of " << nums[0] << " and " << nums[1] << " is " << sum\_result << endl;  return nullptr;  }  int main() {  pthread\_t threads[3];    // Arguments for each thread  int fact\_num = 5; // Factorial of 5  int fib\_num = 10; // Fibonacci series up to 10  int nums[2] = {3, 7}; // Sum of 3 and 7  // Create threads  pthread\_create(&threads[0], nullptr, factorial, &fact\_num);  pthread\_create(&threads[1], nullptr, fibonacci, &fib\_num);  pthread\_create(&threads[2], nullptr, sum, nums);  // Wait for all threads to finish  for (int i = 0; i < 3; ++i) {  pthread\_join(threads[i], nullptr);  }  return 0;  } |

**Question # 5**

|  |
| --- |
| Run this code in Ubuntu, Understand it and check the output. |
| #include <iostream>  #include <pthread.h>  using namespace std;  // Structure to hold arguments for the operations  struct OperationArgs {  int a;  int b;  };  // Function to perform addition  void\* add(void\* arg) {  OperationArgs\* args = static\_cast<OperationArgs\*>(arg);  int result = args->a + args->b;  cout << "Sum of " << args->a << " and " << args->b << " is " << result << endl;  return nullptr;  }  // Function to perform subtraction  void\* subtract(void\* arg) {  OperationArgs\* args = static\_cast<OperationArgs\*>(arg);  int result = args->a - args->b;  cout << "Difference of " << args->a << " and " << args->b << " is " << result << endl;  return nullptr;  }  // Function to perform multiplication  void\* multiply(void\* arg) {  OperationArgs\* args = static\_cast<OperationArgs\*>(arg);  int result = args->a \* args->b;  cout << "Product of " << args->a << " and " << args->b << " is " << result << endl;  return nullptr;  }  int main() {  pthread\_t threads[3];  OperationArgs args;  // Initialize the operands  args.a = 8; // First number  args.b = 4; // Second number  // Create threads for each operation  pthread\_create(&threads[0], nullptr, add, &args);  pthread\_create(&threads[1], nullptr, subtract, &args);  pthread\_create(&threads[2], nullptr, multiply, &args);  // Wait for all threads to finish  for (int i = 0; i < 3; ++i) {  pthread\_join(threads[i], nullptr);  }  return 0;  } |

**Note:**

For question 4 and 5 just write the output of code and a short note what does this code is doing.