Name:- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Answer the following Questions

**Q1: [5 points]** Multithreading is supposed to reduce the time of execution. Practically, when multithreading is applied on a small data set, utilizing more threads increase the execution time instead of reducing it, why?

|  |
| --- |
| Use this space to answer the question  The data in the memory of one thread can start to overwrite the data of other threads. |

**Q2: [6 points programming Question]** write 2 C++ methods to compute the dot product of two vectors. the first method will use a regular iteration process to complete the calculation, the second method will use the parallelism concept using openMP. Compile and run your project and provide the results.

|  |
| --- |
| Use this space to answer the question  *#include* <iostream>  *#include* <vector>  *#include* "omp.h"  *#define* size 4  using namespace std;  int Dot\_Product\_Reg(vector <int> vect\_a, vector <int> vect\_b) {      int prod;      prod = 0;  *for* (int i = 0; i < size; i++) {          prod += vect\_a[i] \* vect\_b[i];      }  *return* prod;  }  int Dot\_Product\_Par(vector <int> vect\_a, vector<int> vect\_b) {      int product = 0;  *#pragma* omp parallel num\_threads(3) for  *for* (int i = 0; i < size; i++)          {              product += vect\_a[i] \* vect\_b[i];          }  *return* product;  }    int main() {      vector <int> vector\_a = {5, 8, 4, 3};      vector <int> vector\_b = {6, 3, 1, 2};      cout << "Dot product regular:" << endl;      cout << Dot\_Product\_Reg(vector\_a, vector\_b) << endl;      cout << "Dot product with openMP" << endl;      cout << Dot\_Product\_Par(vector\_a, vector\_b) << endl;  *return* 0;  }  Dot product regular:  64  Dot product with openMP  64 |

**Q3: [6 points]** Check for balanced parentheses in an expression using **a stack**. Given the expression string below, examine whether the pairs and the orders of “{“,”}”,”(“,”)”,”[“,”]” are correct. Show the stack each time it is modified. Show the changes applied to the stack with every character you read from the expression. **Note: This is not a programming question.**

a\*(b-c)/{[3-d]\*(4+9)}

|  |
| --- |
| Use this space to answer the question  (  none  {  {[  {  {(  {  done |

**Q4: [5 points]** Trace the evaluation of the following expression. Show the operand **stack** each time it is modified

**Note: This is not a programming question.**

6 14 + 5 \* 19 45 5 / - \*

|  |
| --- |
| Use this space to answer the question  6  6, 14  20  20, 5  100  100, 19  100, 19, 45  100, 19, 45, 5  100, 19, 9  100, 10  1000 |

**Q5: [3 points programming Question]** by using the STL, complete the following tasks: By using a stack, write a function to insert an element in the middle of a stack.

You are asked to post the code and provide the results of running your project for the function

|  |
| --- |
| Use this space to answer the question  *#include* <iostream>  *#include* <stack>  using namespace std;  int main() {      stack<int> stack, stack\_new;      int size;      stack.push(36);      stack.push(35);      stack.push(34);      stack.push(33);      size = stack.size();      cout << "Original Stack" << endl;  *for* (int i = 0; i < size; ++i) {          cout << stack.top() << endl;  *if* (i == size/2) {              stack\_new.push(32);          }          stack\_new.push(stack.top());          stack.pop();      }      size = stack\_new.size();      cout << "New Stack" << endl;  *for* (int i = 0; i < size; ++i)      {          cout << stack\_new.top() << endl;          stack\_new.pop();      }  *return* 0;  }  Original Stack  33  34  35  36  New Stack  36  35  32  34  33 |