Education is the manifestation of perfection already in man.
## Contents

### Part One

1. **Overview** .................................................. 7

1.1 Salient Features of the Department .......................... 8

1.2 Program Requirements ........................................ 9

1.2.1 Semester Calendar Credit Hours ........................ 9

1.2.2 Quarter Calendar Credit Hours ............................ 11

1.2.3 Attendance .................................................. 12

1.3 Faculty ....................................................... 12
# Part Two

## 3 Course Syllabus ........................................... 19

3.1 Course List ........................................ 20
3.2 Course Description .................................... 21
3.2.1 Theoretical Computer Science ................. 21
3.2.2 Algorithms ....................................... 23
3.2.3 Programming and Systems ....................... 26
3.2.4 Project .......................................... 31

## Index ......................................................... 32
1. Overview

Computer Science plays a pivotal role in this era of digital revolution. Its presence in today’s world is ubiquitous. There is a great demand, both in industry and in academia, for computer science graduates who have sound knowledge of the underlying concepts and principles governing computer science as well as the ability to apply them to solve real-world problems.

The Department of Computer Science at Ramakrishna Mission Vivekananda University trains students to achieve both these goals, which will enable them to excel in their professional careers - whether in academics, research or industry.

Any graduate holding a bachelor degree in any field from any university is supposed to have a broad knowledge of that field. A master
degree holder on the other hand must have a deeper understanding of the same broad subjects as well as some depth of knowledge in a chosen field. A doctorate in contrast indicated that one has contributed to a field of knowledge in some original way.

The programme thus aims to make a graduate student suitable for both research as well as software industry. After the successful completion of the programme one should have sufficient breadth of knowledge to be absorbed in any applied software industry, and, if one chooses the path of academic career, to do research in the selected areas in which one has acquired some depth.

1.1 Salient Features of the Department

- Highly qualified & dedicated teaching faculty
- Excellent student-faculty ratio (4:1)
- Regularly revised curriculum based on industry trends and experts feedback. (for eg. inclusion of Android programming, Machine Learning, Data Mining courses etc.)
- Three Computer Labs with state-of-the-art desktops (8-core CPU & 8GB RAM)
  - Database Lab
  - Pattern Recognition & Machine Learning Lab
  - Post-Graduate Research Lab
- Conferences and workshops regularly organized/attended by the faculty
- Industry-grade servers to run university’s websites, mail-server, firewall etc.
- Few of the courses are taught at ISI, Kolkata
- Member of National Knowledge Network (high speed internet - 1 Gbps)
- Individual email accounts and homepages for students
1.2 Program Requirements

The student is supposed to complete the courses requirement, the project requirement, the seminar requirement and the thesis requirement.

Maximum no. of breadth and background courses credit that will be counted towards the degree is 24. Minimum no. of depth courses credit required for the degree is 20. A student is required to finish a minimum of 72 credits to be eligible for the degree which include background courses, breadth courses, depth courses, project and research as applicable.

The courses are divided in three categories viz., Theoretical computer science, Algorithms, Programming and systems, and a Project. We take into account the recommendations of board of studies in setting up the courses for each semester and the syllabus to be taught. In addition to this the instructor has fair amount of freedom to improvise by including advanced content at the exclusion of certain preliminary ones depending on the students’ ability to grasp cope with intensity.

1.2.1 Semester Calendar Credit Hours

Most higher education institutions operate on an academic year divided into two equal semesters of 15-16 weeks duration, with a winter break
of 2-3 weeks and a summer session of 10-12 weeks, plus additional shorter breaks. The actual amount of academic work that goes into a single semester credit hour is often calculated as follows:

One lecture (taught) or seminar (discussion) credit hour represents 1 hour per week of scheduled class/seminar time and 2 hours of student preparation time. Most lecture and seminar courses are awarded 3 credit hours. Over an entire semester, this formula represents at least 45 hours of class time and 90 hours of student preparation. One laboratory credit hour represents 1 hour per week of lecture or discussion time plus 1-2 hours per week of scheduled supervised or independent laboratory work, and 2 hours of student preparation time. Most laboratory courses are awarded up to 4 credit hours. This calculation represents at least 45 hours of class time, between 45 and 90 hours of laboratory time, and 90 hours of student preparation per semester for a 3 credit laboratory course. For a pure 3 credit laboratory course, this may amount to between 90 to 135 hours of laboratory time and 90 hours of student preparation time. One practice credit hour (supervised student teaching, field work, supervised clinical rounds, visual or performing art studio, etc.) represents 3-4 hours per week of supervised and/or independent practice. This in turn represents between 45 and 60 hours of work per semester. Blocks of 3 practice credit hours, which equate to a studio or practice course, represent between 135 and 180 total hours of academic work per semester. One independent study (thesis or dissertation research) hour is calculated similarly to practice credit hours. Internship or apprenticeship credit hours are determined by negotiation between the super- visiting faculty and the work supervisor at the cooperating site, both of whom must judge and certify different aspects of the students work. The credit formula is similar to that for practice credit. This masters degree program requires at least 33 (normally 35-40 i.e. 10 courses ) credit hours and including a research thesis or project (to make up 60 credits) represents over 4,000 actual hours
1.2 Program Requirements

of supervised and unsupervised (independent research) study. To summarise, one year the Msc programme requires at least 30 credit hours which for usual lecture type courses translates to minimum 450 hours of lectures including 900 hours of student preparation. If some of the courses are practical type courses (1 lectures + 1-2 practical) than also the credits are determined by the number of lectures. If a course is only practical, field work or independent study type then a three-four hour lab may earn a single credit, a six-eight hour lab may earn two credits.

1.2.2 Quarter Calendar Credit Hours

This department has adopted in the past a quarter calendar, in which the academic year is divided into three terms, called quarters, of 10-11 weeks duration plus a summer session (considered the fourth quarter, but optional), a short winter term and other calendar breaks. Quarter credit hours represent proportionately less work than semester hours due to the shorter terms, about two-thirds of a semester credit hour (i.e. 1 quarter hour is 10 hours of lecture and 20 hours of student preparation). Thus, a bachelors degree on the quarter calendar may require a minimum of 180 quarter hours, which compares to 120 semester hours. A masters degree on the quarter calendar may require a minimum of 90 quarter hours. So, even if the academic calendar is quarterly, the number of lecture hours and student preparation times are same. Courses taught The courses that are taught in the curriculum largely depends on what is decided by the Board of Studies in a meeting that is convened before every semester begins. We typically offer theoretical courses that help the student to pursue with research. Technology based courses are also offered taking students preferences into consideration. The whole list of courses that were offered in the Msc programme over the last 8 years is summarised in the next section.
1.2.3 **Attendance**

A student is expected to have at least 80 percent attendance in all the courses he has enrolled in.

1.3 **Faculty**

List of faculty members in the department of computer science and their field of specialization.

<table>
<thead>
<tr>
<th>Name</th>
<th>Areas of Interest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swami Sarvottamananda, [PhD] (course coordinator)</td>
<td><em>Computational Geometry</em></td>
</tr>
<tr>
<td>Swami Dhyanagamyananda, [Mtech CS]</td>
<td><em>Graph Theory</em></td>
</tr>
<tr>
<td>Prof. Subir Ghosh, [PhD]</td>
<td><em>Algorithms, Computational Geometry</em></td>
</tr>
<tr>
<td>Br. Vikas, [PhD]</td>
<td><em>Image Processing, Pattern Recognition, Computer Vision</em></td>
</tr>
<tr>
<td>Br. Sourav Pradhan, [MTech CS]</td>
<td><em>Computational Geometry, Data structures</em></td>
</tr>
</tbody>
</table>
2. Admissions

Admissions for MSc in Computer Science at Belur Campus.

2.1 Eligibility to Apply

Students of BSc Hons or BTech in computer science/engineering, BTech in any other discipline with a Diploma in Computer Science, or Bachelor in Computer Applications from any recognized University/Institute with 60% marks in aggregate (in the case of Hons students: 60% in Hons subject) will be eligible to apply.

Admission for MSc Computer Science in Belur campus is open for male candidates only.
2.2 How to Apply

Application forms and Admit Card can either be downloaded from the Department’s website or collected from the University’s central office at Belur (just outside the Belur Math main gate). Please read the instructions printed in the application form carefully. All the relevant information of the application must be duly filled and submitted along with a fee of Rs.300/-. It can be either a demand draft for Rs 300/- made in favour of Ramakrishna Mission Vivekananda University payable at Belur Math, Howrah, (preferably UBI or SBI) or by cash. Completed application forms can be submitted in any of the following ways as given below.

1. Send the completed application form with the demand draft to the following address on/before the last date

Registrar
Ramakrishna Mission Vivekananda University
PO Belur Math
Howrah, West Bengal 711202

2. Submit the completed form in person with the demand draft or cash to the University’s central office at Belur on/before the above-mentioned last date.

3. Send the completed application preferably in pdf format by email to: cs[at]rkmu[dot]ac[dot]in. Bring a printout of the completed application form and printout of the email sent to us along with the demand draft or cash to the University’s central office at Belur on the day of the entrance test at least half hour before the commencement of the admission test. Applicants have to duly fill the admit card details and bring it on the day of admission test. Please do not submit the admit card along with your application.
2.3 Fee Structure

Admission process shall begin immediately after the interview. Those candidates permitted and willing to take admission may get admitted immediately by paying a total amount of Rs 17,050/- having the following breakup.

- Tuition fee - Rs 6000/- for the first semester plus
- Admission cum registration fee (one time)- Rs 5000/-
- Lab fee - Rs 3000/-
- Examination fee - Rs 1000/-
- Id Card - Rs 50/-
- Caution Deposit (one time, refundable) - Rs 2000/-
- Fee for the subsequent semester (i.e., 2nd, 3rd, 4th ) is Rs.10,000/- each( consisting of Tuition fee Rs.6000/-, Lab fee Rs. 3000/-, Exam fee Rs.1000/-)

Selected candidates have to take admission by paying the requisite fees. Those who fail to take admission before the stipulated date without explicit permission from the authority will run the risk of forfeiting their admission.

For the entire period of 2 years, the student will need to pay as fees Rs.47,080/-
Part Two

3  Course Syllabus  ........................................ 19
   3.1  Course List
   3.2  Course Description

Index  .......................................................... 32
3. Course Syllabus

The courses are mainly drawn from the branch of theoretical Computer Science. A few application based courses too have been included with a view to cater to the industry requirements and trends. A list of courses offered in the MSc Computer Science programme is given below. For a detailed listing of the topics covered under each course, see Section 3.2
# Chapter 3. Course Syllabus

## 3.1 Course List

### Theoretical Computer Science

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS200</td>
<td>Automata Theory</td>
</tr>
<tr>
<td>CS201</td>
<td>Discrete Mathematics and Logic</td>
</tr>
<tr>
<td>CS205</td>
<td>Introduction to Probability</td>
</tr>
<tr>
<td>CS300</td>
<td>Theory of NP-Completeness</td>
</tr>
<tr>
<td>CS301</td>
<td>Computational Complexity</td>
</tr>
<tr>
<td>CS206</td>
<td>Probability and Stochastic processes</td>
</tr>
</tbody>
</table>

### Algorithms

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS110</td>
<td>Analysis of Algorithms</td>
</tr>
<tr>
<td>CS211</td>
<td>Graph Algorithms</td>
</tr>
<tr>
<td>CS212</td>
<td>Computational Geometry</td>
</tr>
<tr>
<td>CS214</td>
<td>Computer Graphics(*)</td>
</tr>
<tr>
<td>CS222</td>
<td>Introduction to Discrete Optimization</td>
</tr>
<tr>
<td>CS310</td>
<td>Advanced Data structures</td>
</tr>
<tr>
<td>CS312</td>
<td>Approximation and Online algorithms</td>
</tr>
</tbody>
</table>

### Programming and Systems

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS123</td>
<td>Concepts of Programming Languages</td>
</tr>
<tr>
<td>CS125</td>
<td>Programming Skill Development Project</td>
</tr>
<tr>
<td>CS220</td>
<td>Object Oriented Programming using Design Patterns</td>
</tr>
<tr>
<td>CS221</td>
<td>Compiler Design</td>
</tr>
<tr>
<td>CS222</td>
<td>Theory of Operating systems</td>
</tr>
<tr>
<td>CS250</td>
<td>Design and Implementation of Database Management System</td>
</tr>
<tr>
<td>CS229</td>
<td>Android Programming for Handheld Devices</td>
</tr>
<tr>
<td>CS321</td>
<td>Distributed Computing Systems</td>
</tr>
<tr>
<td>CS234</td>
<td>Computer Architecture</td>
</tr>
</tbody>
</table>
3.2 Course Description

Project

CS404 Independent/Group (of two) Project

Please Note:

*Though all these courses have been taught in the past, this is only a tentative list and may change for the current academic year. The Department takes feedback from various sources (eg. Board of Studies, students etc.) before finalising the courses for a given semester.*

### 3.2 Course Description

The syllabus of each of the courses is as follows.

#### 3.2.1 Theoretical Computer Science

**CS200 Automata Theory**

Theory of Computers. Topics: Regular sets, finite automata, regular expressions, equivalences among notations, methods of proving a language not to be regular. Context-free languages, grammars, pushdown automata, normal forms for context-free grammars, proving languages non-context-free. Turing machines, equivalent forms, undecidability. Nondeterministic Turing machines, their properties, the class NP, complete problems for NP, Cook’s theorem, reducibilities among problems. Programming assignments

Reference:

1. Introduction to the Theory of Computation by Sipser
2. Introduction to Automata theory, languages, and computation by Hopcroft, Ullman, Motwani
3. Automata theory by Sakrovitch
4. An introduction to formal languages and automata by Peter Linz

Credit: 5, Prerequisite: Discrete Maths.
**CS201 Logics for Computer Science**

An elementary exposition from a computational point of view of propositional and predicate logic, axiomatic theories, and theories with equality and induction.

Topics: Interpretations, models, satisfiability, validity, proof strategies and applications. Automated deduction: skolemization, unification, resolution, equality.

Temporal Logics, LTL, CTL, Model checking, Program Verification, Binary Decision Diagrams,

Reference:
1. Logic for Computer Scientists by Uwe Schöning
2. Logic in Computer Science: Modelling and Reasoning about systems by Huth and Ryan

**CS205 Introduction to Probability**

For studying randomized and approximation algorithms we need to know a good amount of probability and this course is fashioned to serve this need.


Reference:
1. Introduction to Probability, Charles M. Grinstead Credit 3.

**CS300 Theory of NP-completeness**
3.2 Course Description

Topics: Polynomial time Algorithms and Intractable problems, Decision problems, deterministic turing machines and the Class P, Non-deterministic computation and the class NP, relation between P and NP, Polynomial Transformation and NP-completeness, Cook’s theorem, techniques for proving NP-completeness.

Reference:
1. Computers and Intractability by Garey and Johnson
2. Computational Complexity by Papadimitrou

Credit: 5, Prerequisite: Automata Theory

**CS301 Complexity Theory**

Topics: Basic resources for computation (time, space, nondeterminism) and their associated complexity classes (P, NP, PSPACE and more). Relationships among resources (P versus NP, time versus space, and more). Reductions & completeness (NP completeness, PSPACE completeness, and more. Counting problems, #P. Randomness as a computational resource; associated complexity classes. Nonuniform models of computation; circuit complexity; lower bounds. Communication complexity. Interactive proofs & IP=PSPACE. Probabilistically checkable proofs (PCP) and inapproximability.

Reference:
2. Computational Complexity by Papadimitrou

Credit: 4, Prerequisite: Theory of NP-completeness

3.2.2 Algorithms

**CS110 Analysis of Algorithms**
Chapter 3. Course Syllabus

Topics: Elementary Data structures, Recursion, Permutations and Combinations, Basic number theory, Euclid algorithm, Chinese Remainder Theorem, Elementary Probability theory, Bayes theorem, O and Omega notation, Master theorem, review of sorting algorithms, Medians and order statistics, Graph algorithms - BFS, DFS, Prim, Kruskal and Dijkstra algorithms, connected components.

Reference:
1. Introduction to Algorithms by Cormen et al
2. Lecture notes by David Mount and David Eppstein
3. Algorithms in C++ by Sedgewick

Credit: 4, Prerequisite: Knowledge of a Programming Languages and Discrete Maths.

CS211 Graph Algorithms


Credit: 3, Prerequisite: Discrete Maths and Programming Techniques.

CS212 Computational Geometry

Topics: Geometric primitives, Convex Hulls (Graham’s scan, Jarvis’ March, Chan’s algorithm), Plane Sweep and Incremental Construction paradigms, Point Location and trapezoidal maps, triangulation, line segment intersections, kd-trees, Binary Space Partition trees, Voronoi diagrams and Delaunay triangulations, Introduction to duality, Art Gallery theorem, Planar graphs, Euler’s theorem and applications, Arrangements of lines and points, Zone theorem. Program-
CS213 Computer Graphics

Reference:
1. Computer Graphics by Hearn and Baker
2. Lecture notes by David Mount
3. Lecture Notes by Shene

CS310 Advanced Data Structures

Reference:
1. Advanced Data Structures by Peter Brass
2. Introduction to Algorithms by Cormen et al
3. Lecture notes by David Mount
4. Algorithms in C++ by Sedgewick

CS312 Approximation and Online Algorithms
Introduction: 1. Background, 2. Performance Measure


References:

3.2.3 Programming and Systems

CS123 Concepts of Programming Languages

Imperative, Object-oriented and Functional Programming. Advanced programming language topics

Topics: Syntax and Semantics, Algebraic specification of Types, Advanced memory management features of C and C++; the differences between imperative and object-oriented paradigms. The functional paradigm (using Scheme, a dialect of LISP) and concurrent programming (using C and C++). Brief survey of other modern languages such as Python, Perl, Tcl/Tk, C#, and Prolog.

Customised new; manual destruction; operator overloading; STL template containers; algorithms; iterators; single and multiple inheritance; class hierarchy design; and C++ pitfalls.

Programming in Java. Topics: OOP design, design patterns, test-
3.2 Course Description

Design principles of programming languages, C++: An Object-Oriented Extension of C, Syntax Lexical Structure of Programming Languages, CFG and BNF, Parse Trees and Abstract Syntax Trees, Ambiguity Associativity and Precedence, parsing semantics, Symbol table, name resolution, overloading, aliases and dangling references, Data Types, Algebraic specification of abstract data, expressions and their construction, exception handling, memory management Advanced topics. Functional programming, delayed evaluation, lambda calculus, Logic programming, parallel programming. Packages, programming ML Threads, Semaphores, Monitors, Message Passing, Parallelism in Non-Imperative Languages

References:
1. Concepts in Programming Languages by John C. Mitchell
2. Essentials of Programming Languages by Friedman, and Wand
3. Programming Languages: Principles and Practice by Kenneth Louden
4. Programming Language Pragmatics by Michael Scott

Credit: 4, Prerequisite: None

**CS125 Programming Skill Development Project**

This is a lab oriented course, offered during summer for two months. The students are given around 15 programming assignments drawn from different fields of computing. The students are also given the choice of executing a mini-project in the subject domain of their preference in consultation with the course coordinator. Credits: 3, Prerequisite: Programming techniques, Algorithms. CS220 Design Patterns using C++ - Introduction SOLID principles of OO Software Design, Motivation for design patterns, Creational patterns, Struc-
tural patterns, Behavioral Patterns, Basics of UML.

Reference:
1. Design Patterns by Gamma et al
Credits: 3

CS221 Compiler Design
Principles and practices for design and implementation of compilers and interpreters. Topics: lexical analysis; parsing theory; symbol tables; type systems; scope; semantic analysis; intermediate representations; runtime environments; code generation; and basic program analysis and optimization. Students construct a compiler for a simple procedural language during course programming projects.

References:
1. Compilers Principles, Techniques, & Tools by Aho, Sethi, Ullman, Monica
2. Engineering a Compiler by Keith D Cooper, Linda Torczon
3. Modern Compiler Implementation in Java by Andrew W. Appel
Credit: 4, Prerequisite: Algorithms and Automata Theory.

CS222 Theory of Operating systems
Topics: Basic structure of an operating system; synchronization and communication mechanisms; implementation of processes, process management, multi-threading, scheduling, and protection; memory organization and management, including virtual memory; I/O device management, secondary storage, and file systems.

Reference
1. Modern Operating Systems by Tannenbaum
Credit: 4, Prerequisite: C Programming Language.

CS250 Design and Implementation of Database Management System
The main focus of this second level course on Database Systems shall be internals of a RDBMS. Lecture topics include the following:

1. **Storage and Indexing**
   - Query Processing Algorithms, Alternatives for performing selection
   - External sorting - run generation using replacement selection
   - Joins (nested loops, merge, hash, hybrid hash)
   - Grouping and aggregation - Other set operations

2. **Query Optimization**
   - Equivalence rules to enable query transformation
   - Structure of a cost-based query optimizer, statistics on base relations, estimating stats of intermediate results, join enumeration, dynamic programming for join order optimization

3. **Transactions**
   - ACID properties and their need
   - Need for concurrent execution
   - Schedules and notion of serializability
   - Conflict and view serializable schedules
   - Testing for serializability
   - Isolation levels in ANSI SQL

4. **Concurrency Control**
   - Lock based protocols (2 phase locking), recoverable and cascade less schedules
   - Deadlock avoidance and resolution protocols, tree protocol, multi granularity locking
   - Time permitting we shall cover time-stamp based protocols and MVCC

**References:**

1. Fundamentals of Database Design by Elmasri and Navathe
2. Database Management systems by Raghu Ramakrishnan

Credit: 4, Prerequisite: Algorithms.

**CS229: Android Programming for Handheld Devices**

Topics: Android Fundamentals & Overview, Dalvik Virtual Machine (DVM), Android Internals, Application Runtime Environment, GUI Architecture, Activity, BroadcastReceivers, Services, Content Providers, Intents, Manifest file, UI Components, Widgets, Event Receivers, Views, Dynamic UIs, Media Player, SDCARD (Read & write), SQLite Database, File handling, Shared Preference, Notification, Android Timer, Handler, Thread, Android ProgressBar,
Chapter 3. Course Syllabus


Reference:

Credits: 6, Prerequisite Java programming.

CS321 Distributed Computing Systems

Distributed file system case studies, low-level database storage techniques, and distributed programming, some distributed algorithms. Topics: File system structures, journalling and logging, I/O system performance, Distributed Objects and invocation, remote procedure call abstraction, and systems illustrating these concepts, Java RMI, CORBA. Replication and consistency, Security, Leader election, Distributed deadlock detection, Distributed name service, fault tolerance, and crash recovery, Programming assignments.

Reference:
1. Distributed Systems: Concepts and Design by Coulouris et al
2. Distributed Systems: Principles and Paradigm by AS Tannenbaum
3. Java Network Programming and Distributed Computing by David Reilly, Michael Reilly

Credit: 4, Prerequisite: Algorithms

CS323 Discrete Event Systems

Systems and Models, Supervisory Control, Petri Nets, Timed Automata. Topics: Systems and Control basics, Goal of system theory, Feedback control with supervisors, Control with partial controllability, Non-blocking control, Control with modular specifications, Con-
3.2 Course Description

trol with partial observation, Decentralized control, Basics-Analysis-
and-control of Petri Nets, Comparison of Petri Nets with automata, 
Timed Automata.

Reference:
1. Introduction to Discrete Event Systems, Christos G. Cassandras, 
Stéphane Lafortune
Credit: 4, Prerequisite: Automata Theory

CS222 Introduction to Discrete Optimization

Linear Programming (LP), weak and strong duality theorems, the 
simplex algorithm. Concepts of duality, shadow price (simplex mul-
tiplier/dual price), interpretation of shadow price, complementary 
slackness, reduced cost. Problem formulations and use of software 
to solve LP problems. Integer programming, formulations and solu-
tion methods including the branch and bound procedure. Transporta-
tion problem.

Introduction to non linear programming.

Important concepts will be covered primarily based on the below 
reference.

Reference:
1. Applied Mathematical Programming by Bradley, Hax and Mag-
nanti

3.2.4 Project

CS404 Independent Project

Group or individual projects under faculty direction. A project can 
be either a significant software application or publishable research. 
Software application projects include substantial programming and 
modern user-interface technologies and are comparable in scale to
shareware programs or commercial applications. Research projects may result in a paper publishable in an academic journal or presentable at a conference. Required public presentation of final application or research results.

Credit: 8, Prerequisite: Permission of the faculty supervisor.