ACADEMIC CHALLENGE FOR ACES ENGINEERING AND SCIENCE

EASTERN ILLINOIS UNIVERSITY

2023 Academic Challenge

STATE COMPUTER SCIENCE EXAM

Computer Science Test Production Team

Andrew Ian Chen, Independent Contractor – Author, Reviewer
Stephanie Akemi Brandt, Northrup Grumman Corporation – Reviewer

GENERAL DIRECTIONS

Please read the following instructions carefully. This is a timed test; any instructions from the test supervisor should be followed promptly.

The test supervisor will give instructions for filling in any necessary information on the answer sheet. Most Academic Challenge sites will ask you to indicate your answer to each question by marking an oval that corresponds to the correct answer for that question. One oval should be marked to answer each question. Multiple ovals will automatically be graded as an incorrect answer.

Be sure ovals are marked as ● , not ⊙ , ⊕ , ◑ , etc.

If you wish to change an answer, erase your first mark completely before marking your new choice.

You are advised to use your time effectively and to work as rapidly as you can without losing accuracy. Do not waste your time on questions that seem too difficult for you. Go on to the other questions, and then come back to the difficult ones later if time remains.

Time: 40 Minutes Number of Questions: 30

DO NOT OPEN TEST BOOKLET UNTIL YOU ARE TOLD TO DO SO!

2023 Academic Challenge
Computer Science Sectional Exam

For each question select the best option. Assume required header files and libraries are included for example programs.

1. Which of the following operating systems uses a monolithic kernel?
    a. Windows NT
    b. Linux
    c. macOS
    d. Minix
    e. ReactOS

2. Which of the following data structures is not implemented as a type of tree?
    a. BST (for binary search)
    b. Heap
    c. Hash table
    d. Trie
    e. Treap

3. Which of the following sorts arranges a set of ordinal elements in ascending order by repeatedly finding the minimum element and swapping it with the first unsorted element?
    a. Insertion sort
    b. Selection sort
    c. Bubble sort
    d. Merge sort
    e. Quick sort

4. Which term best describes device that connects multiple devices on a network and groups data packets transmitted on the network?
    a. Switch
    b. Repeater
    c. Bridge
    d. Firewall
    e. Protocol converter

5. Which feature of object-oriented programing allows a class to retain attributes and methods from another class?
    a. Inheritance
    b. Encapsulation
    c. Abstraction
    d. Polymorphism
    e. Method overriding

6. Which standard defines the request-response protocol for web-server/browser-client interactions?
    a. TCP
    b. HTML
    c. WWW
    d. URL
    e. HTTP

7. Which parallel processing architecture achieves parallelism via asynchronous and independently running processors?
    a. Vector processor (VP)
    b. Single instruction, multiple data (SIMD)
    c. Single instruction, single data (SISD)
    d. Multiple instruction, single data (MISD)
    e. multiple instruction, multiple data (MIMD)

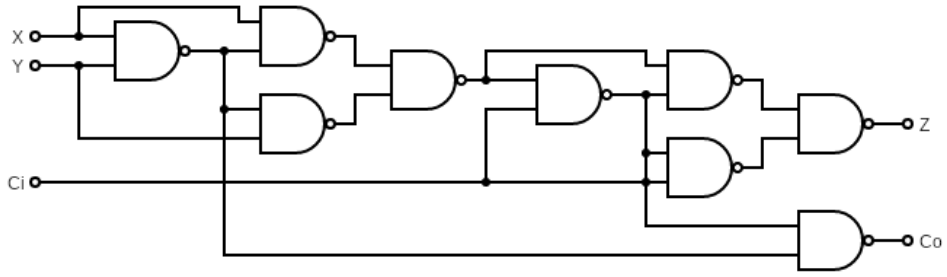
8. Which of the computer file abstraction is not handled by the runtime operating system level?
    a. Virtual file system
    b. Logical file system
    c. Physical file system
    d. fstab
    e. Abstract file system

9. Which of the following will produce the logic table below?

A	B	C	Output
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

    a. $A \cdot \bar{B} + C \cdot A$
    b. $A \cdot B + B \cdot (C \cdot C + C \cdot C)$
    c. $\bar{C} \cdot B + A \cdot (B \cdot B + A \cdot B)$
    d. $(C \cdot C + \overline{A \cdot C}) \cdot \bar{0} + B \cdot \bar{A}$
    e. $A \cdot (\overline{A \cdot B} + C \cdot B) + C$

10. Assume that X and Y are inputs, and Z is an output. Ci and Co are input and output control or indicator bits. What circuit does the following diagram represent?



    a. Adder-subtractor
    b. Delta-sigma modulation
    c. Carry-lookahead adder
    d. Half adder
    e. Full adder

11. What decimal number results from the addition of the hexadecimal numbers: 0x8ABF + 0xB781?
    a. 8781
    b. 47807
    c. 14240
    d. 82496
    e. 241100

For Questions 12 and 13 consider the following program:

```
00  #include <iostream>
01  #include <string>
02  using namespace std;
03
04  int xs3_2_dec(string xs3)
05  {
06      int dec = 0;
07      for (int i = 0; i < xs3.length(); i += 4) {
08          int bit = 0;
09          for (int j = 0; j < 4; j++) {
10              bit *= 2;
11              bit += xs3[i+j] - '0';
12          }
13          dec *= 10;
14          dec += bit - 3;
15      }
16      return dec;
17  }
18
19  int main()
20  {
21      string xs3 = "10101000";
22      int dec = xs3_2_dec(xs3);
```

```
23      cout << "dec:  " << dec << endl;
24      return 0;
25  }
```

12. What is the output of the program?
    a. dec: 108
    b. dec: 75
    c. dec: 810
    d. dec: 57
    e. dec: 168

13. What is the type of encoding is the Excess-3 (xs3) encoding implemented in the program?
    a. American Standard Code for Information Interchange(ASCII)
    b. Double dabble
    c. Binary-coded decimal (BCD)
    d. Hamming code
    e. Message-digit algorithm

14. If x = 10, what is the result of the operations “x << 2 | 32”?
    a. 1
    b. 32
    c. 40
    d. 34
    e. 0

15. The prefix increment operator (e.g. ++x) has less precedence relative to the _____operator?
    a. Addition (+)
    b. bitwise XOR (^)
    c. Logical OR (||)
    d. Pointer to member (->)
    e. Postfix increment (x++)

16. Which of the following is not a reserved keyword in C++20?
    a. this
    b. class
    c. break
    d. function
    e. nullptr

For questions 17 and 18 consider the following program:

```
00  #include <iostream>
01
02  int main()
03  {
04      int x = 14;
05      int y = 28;
```

```
06      for (int i = 1; i < 4; ++i) {
07          x ^= y; // What type of operator?
08          y--;
09      }
10      std::cout << x << std::endl;
11      return 0;
12  }
```

17. What is the output of the program?
    a. 24
    b. 25
    c. 10
    d. 31
    e. 19

18. The operator `^=` on commented line 7 is best described as:
    a. Logical comparison
    b. Member access
    c. Conditional comparison
    d. Arithmetic assignment
    e. Compound logical assignment

For questions 19, 20, and 21, consider the following program:

```
00  #include <iostream>
01  #include <vector>
02  using namespace std;
03
04  const int N = 5;
05  vector<int> graph[N]; // adjacency list
06  int colors[N];
07  bool used[N];
08
09  int get_unused_color()
10  {
11      for (int i = 1; ; i++) {
12          if (!used[i]) {
13              used[i] = true;
14              return i;
15          }
16      }
17  }
18
19  void color_graph()
20  {
21      for (int i = 0; i < N; i++) {
22          for (int j = 0; j < graph[i].size(); j++) {
23              if (colors[graph[i][j]] != 0) {
24                  used[colors[graph[i][j]]] = true;
25              }
26          }
27          int c = get_unused_color();
```

```
28          colors[i] = c;
29          for (int j = 0; j < graph[i].size(); j++) {
30              used[colors[graph[i][j]]] = false;
31          }
32      }
33  }
34
35  int main()
36  {
37      graph[0] = {1, 2, 4};
38      graph[1] = {0, 2, 3};
39      graph[2] = {0, 1, 3, 4};
40      graph[3] = {1, 2, 4};
41      graph[4] = {0, 2, 3};
42
43      color_graph();
44      cout << "Vertex " << 3 << ": Color " << colors[3] << endl;
45      return 0;
46  }
```

19. What is the run time complexity of the `color_graph` function for a number of vertices N?
    a. O(N)
    b. O(N log(N))
    c. $O(N^2)$
    d. O(1)
    e. O(N!)
20. What technique does the function `color_graph` use?
    a. A* search (A-star)
    b. Branch-cut optimization
    c. Recursion
    d. Dynamic programing
    e. Greedy algorithm

21. What is the output of the program?
    a. Vertex 3: Color 0
    b. Vertex 3: Color 1
    c. Vertex 3: Color 2
    d. Vertex 3: Color 3
    e. Run Time Error: index out of bounds

For questions 22, 23, and 24 consider the following program:

```
00  #include <iostream>
01  #include <unordered_map>
02  using namespace std;
03
04  unordered_map<int, bool> Sieve;
05
06  bool Eratosthenes(int n)
07  {
08      if (n == 1 || n == 2) return true;
```

```
09
10      if (Sieve.find(n) != Sieve.end()) {
11          return Sieve[n];
12      }
13
14      bool result = true;
15      for (int i = 2; i * i <= n; i++) {
16          if (n % i == 0) {
17              result = false;
18              break;
19          }
20      }
21      Sieve[n] = result;
22      return result;
23  }
24
25  int main()
26  {
27      Eratosthenes(31);
28      cout << "47: " << Eratosthenes(47) << endl;
29      return 0;
30  }
```

22. What is the output of the program:
    a. 47: 1
    b. 47: 0
    c. 47: 15
    d. 47: 16
    e. 47: Mersenne

23. What is the space complexity in n for the sieve?
    a. O(n log(n))
    b. O(1)
    c. O(log(n))
    d. O(n)
    e. $O(n^2)$

24. What technique is being used in the implementation of the 'Eratosthenes' function?
    a. Coprime factorization
    b. Memoizatoin
    c. Baby-step giant-step
    d. Mersenne cypher
    e. Least squares

For questions 25, 26, and 27 consider the following program:

```
00  #include <iostream>
01  #include <vector>
02  using namespace std;
```

```
03
04  class Shape
05  {
06  public:
07      virtual void draw() const = 0;
08  private:
09      int sides;
10  };
11
12  class Circle : public Shape
13  {
14  public:
15      void draw() const override
16      {
17          cout << "A shape with " << sides << " side: O" << endl;
18      }
19  private:
20      ~Circle() {}
21      int sides = 1;
22  };
23
24  class Square : public Shape
25  {
26  public:
27      void draw() const override
28      {
29          cout << "A shape with " << sides << " side: []" << endl;
30      }
31
32  private:
33      ~Square() {}
34      int sides = 4;
35  };
36
37  int main()
38  {
39      vector<Shape*> shapes;
40      shapes.push_back(new Circle());
41      shapes.push_back(new Square());
42
43      shapes.front()->draw();
44      delete shapes.back();
45      delete shapes.back();
46
47      return 0;
48  }
```

25. What is the output of the program?
    a. A shape with 4 sides: []
    b. A shape with sides:
    c. A shape with 1 side: O
    d. 0
    e. A shape with 1 side: O A shape with 4 sides: []

26. What Object oriented programing technique allows two classes to derive from the Shape class?
    a. Function factories
    b. Functional programing
    c. Polymorphism
    d. Type safety
    e. Heap allocation

27. What error is being made by the program?
    a. `delete` is being called on the same object twice
    b. There is no destructor for objects of type Shape
    c. There is no access to the private destructor for each of the objects
    d. A call to an empty destructor
    e. Assignment to a pure virtual class

For questions 28, 29, and 30 consider the following program:

```
00  #include <iostream>
01
02  class Node
03  {
04  public:
05      int data;
06      Node* next;
07
08      Node(int data)
09      {
10          this->data = data;
11          next = nullptr;
12      }
13  };
14
15  class mysteryStructure
16  {
17  public:
18      Node* head;
19
20      mysteryStructure()
21      {
22          head = nullptr;
23      }
24
25      void addNode(int data)
26      {
27          Node* newNode = new Node(data);
28          newNode->next = head;
29          head = newNode;
30      }
31
32      void print()
33      {
34          Node* temp = head;
35          while (temp != nullptr) {
36              std::cout << temp->data << " ";
```

```
37              temp = temp->next;
38          }
39          std::cout << std::endl;
40      }
41
42      void r()
43      {
44          Node* prev = nullptr;
45          Node* current = head;
46          Node* next = nullptr;
47
48          while (current != nullptr) {
49              next = current->next;
50              current->next = prev;
51              prev = current;
52              current = next;
53          }
54          head = prev;
55      }
56  };
57
58  int main()
59  {
60      mysteryStructure x;
61      x.addNode(5);
62      x.addNode(3);
63      x.addNode(9);
64      x.r();
65      x.print();
66      return 0;
67  }
```

28. What is the output of the program?
    a. 5 3 9
    b. 9 3 5
    c. 5 3 9 9 3 5
    d. 9 3 5 5 3 9
    e. 5 3

29. What is the run time of the function `r` in the number of nodes N?
    a. O(1)
    b. O(log(N))
    c. O(N)
    d. O(N log(N))
    e. $O(N^2)$

30. The list implemented by the '`mysteryStructure`' class is a:
    a. Associative array
    b. Stack
    c. Circular linked list
    d. Doubly linked list
    e. Singly linked list