SCSC012 Test 2: Memorandum with Detailed Explanations

SECTION A

1.1 Uniform Cost Search Tree:

The uniform cost search algorithm expands the least-cost nodes first. Starting at Leesburg, the search would continue to expand neighboring cities (nodes) based on travel costs until Bedford is reached. Each expanded city represents a node in the search tree.

1.2 Heuristic Search Tree for 8-puzzle:

In the 8-puzzle, 'number of tiles out of place' is a simple heuristic. The heuristic evaluates each node by counting how many tiles are not in their goal positions. For the given starting configuration, the algorithm generates a tree based on the potential moves to bring more tiles into the correct position.

1.3 Neural Networks (ANN):

In this case, the processing unit calculates a weighted sum of inputs and passes it through an activation function (such as a threshold or step function). For the output to be 0, the weighted sum must not exceed the activation threshold, which could depend on specific input values and weight settings.

1.4 Basic Components of a Robotic System:

1) Sensors: Devices that detect and measure changes in the environment (e.g., cameras, microphones, etc.).

2) Actuators: These are the motors or mechanisms that allow the robot to interact with and manipulate objects.

3) Control System: The software or hardware responsible for processing sensor inputs and executing actions via actuators.

2.1 Bare Bones Program to Add 1 to X:

A simple Bare Bones program can be written as:

```
clear X;
while X not 0 do;
```

```
    decr X;
    incr X;
end;
```

This program will check if X is not zero, decrement X, then immediately increment it, effectively leaving X unchanged.

2.2 Knapsack Problem Conversion:

Given the magic numbers (642, 2311, 18), we apply a transformation where the easy knapsack problem coefficients are multiplied by the magic numbers, and modulo operations are used to make the problem 'hard'. For instance, calculating each step carefully, we can transform the easy problem into a hard problem.

2.3 Turing Machine Explanation:

The Turing Machine will scan the tape from left to right and replace all consecutive 0s to the left of the '*' with 1s. In each step, the machine moves left or right based on the configuration until it has replaced all the necessary 0s.

3.1 HTML Header Elements:

Each element from a to e represents essential components of an HTML document:

(a) <!DOCTYPE html> - Declaration that defines the document type and version of HTML being used.

(b) <html> - The root element that encapsulates the entire HTML document.

(c) <head> - Contains meta-information (metadata) about the document, like the title and links to stylesheets.

(d) <title> - Specifies the title of the document, shown in the browser's title bar.

(e) <meta> - Provides metadata, such as character set, page description, and viewport settings.

SECTION B

1.1 Debugging C++ Code:

Line 1: Change 'Int' to 'int' because C++ is case-sensitive and requires 'int' for integers.

Line 4: Add a semicolon ';' after declaring 'int num1'. C++ requires a semicolon at the end of variable declarations.

Line 5: Add a semicolon ';' after assigning 'num1 = 4'. This completes the statement.

Line 7: Replace 'adds(4,5)' with 'add(4,5)' to match the correct function name used in line 9.

Line 9: Add 'int' before 'add(int x, int y)' to declare the function's return type (integer).

Line 11: Ensure the return statement 'x + y' is inside the function's curly braces to correctly return the sum.

1.2 Debugging Second C++ Code:

Line 1: Change 'Int' to 'int' for integer return types.

Line 4: Add ',' between 'firstnum' and 'secondnum' to separate variable declarations.

Line 5: Add ';' after 'firstnum = 4' to complete the statement.

Line 6: Replace 'rem(4,5)' with 'remainder(4,5)' to match the function name in line 1.

Line 7: Replace 'adds' with 'remainder' to correctly call the function. The 'remainder' function should return the result of calculating the remainder of the two variables.

Line 9: Ensure 'return firstnum + y' correctly implements the intended remainder logic, which may need fixing if it calculates sum.