Summer Examinations 2021-22

Course Instance Code(s)	2BCT
Exam(s)	B.Sc. in Comp Sc. & Information Technology
Module Code(s)	CT255
Module(s)	NEXT GENERATION TECHNOLOGIES II
Paper No.	1
External Examiner(s)	Dr Ramona Trestian
Internal Examiner(s)	Prof Michael Madden *Dr Michael Schukat *Dr Sam Redfern

Instructions: *Answer one (1) question from section A (20 marks).*
Answer both (2) questions from section B (2x 10 marks).

Duration	2 hours
No. of Pages	10
Discipline(s)	Computer Science
Course Co-ordinator(s)	Dr. Des Chambers

Requirements:

Release in Exam Venue	Yes [✓ No]
MCQ Answer sheet	Yes [✓ No]
Handout	None
Statistical/ Log Tables	None
Cambridge Tables	None
Graph Paper	None
Log Graph Paper	None
Other Materials	None
Graphic material in colour	[✓ Yes] No

PTO

SECTION A
Cyber Security
Answer <u>one</u> question from this section

Q.1.

(i) Assume you develop a database system for your local athletics club that contains information about all club members. Identify and outline 4 GDPR key principles that would inform your design and subsequent use of the system.

[4 marks]

(ii) Discuss in detail three state-of-the-art biometric authentication techniques, their advantages and possible limitations.

[3 marks]

(iii) Using pseudo-code devise an algorithm that calculates the strength of a password, returning a score between 0 (very unsecure) to 5 (very strong). In your algorithm use the strong password characteristics discussed in the lecture, and ignore the problem of readily available information.

[7 marks]

(iv) Devise a simple hash function based on the bitwise EX-OR operation that shows a weak collision resistance. Using an example demonstrate this weakness.

[6 marks]

PTO

Q.2.

(i) Construct an SP network that consists of two parallel 4-bit S-boxes, followed by an 8-bit P-box. Explain in some detail how the input is processed by your network.

[3 marks]

(ii) Rainbow tables are typically used to recover passwords that are hashed once, therefore hashing a password 10 times before storing the resulting value is a good defence mechanism. Explain this statement in some detail and discuss if and how one could enhance rainbow tables to even recover passwords that have been hashed a fixed amount (i.e. 10) times.

[7 marks]

(iii) Combining 2 ciphers may result in a stronger new cipher. Demonstrate this concept by combining the Vigenère cipher with the Rail fence cipher, whereby a plaintext is first encoded using Vigenère, and the resulting ciphertext is subsequently processed using the transposition cipher over 2 rows. Using the keyword "BANANA" as key for the Vigenère cipher, encrypt your surname using the above method.

[7 marks]

(iv) Consider you want to apply steganographic techniques to HTML files, to secretly convey information between a webserver and a client.
Using examples explain in your own words what technique(s) you'd apply and estimate how much information could be hidden. In your solution consider the internal structure of a HTML document.

[3 marks]

PTO

SECTION B
Digital Media – 2D Games in Java
Answer <u>both</u> questions from this section

Q.3.

Consider the simple Java game depicted below. In this game, enemy ‘androids’ move slowly down the screen towards the player. The player moves left and right under control of the arrow keys. The player can also shoot a bullet, by pressing the spacebar. Only one bullet may be on screen at a time, i.e. the player cannot fire again until the current bullet has disappeared off the top of the screen or hit an ‘android’. When an ‘android’ is hit, it resets to the top of the screen.

Carefully read the code provided below, and write appropriate code wherever there is a “**TO DO**” comment. Note that all of your work will be in the main application class (AndroidAttack) – i.e. the three game object classes (Android, Player, Bullet) are already complete. You should make use of the public methods of these classes, but do not alter them.

(i) Write the necessary code to move all game objects in the run() method of the AndroidAttack class

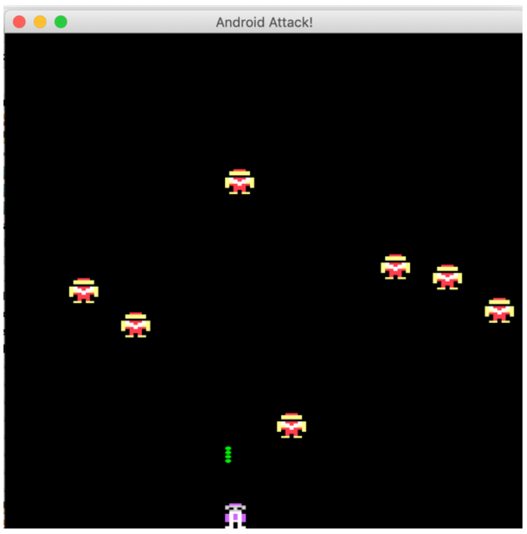
[5 marks]

(ii) Write the necessary code to handle player movement and shooting, in the keyPressed() method of the AndroidAttack class

[3 marks]

(iii) Write the necessary code to stop the player moving, in the keyReleased() method of the AndroidAttack class

[2 marks]



```
public class AndroidAttack extends JFrame implements Runnable, KeyListener {
   // application data
   public static final int NUMANDROIDS = 10;
   private final Dimension screenSize = new Dimension(500,500);
   private static Android[] androidsArray = new Android[NUMANDROIDS];
   private Player player;
```

```java
private Bullet bullet;

// constructor
public AndroidAttack() {
  // load raster graphics and instantiate game objects
  ImageIcon icon = new ImageIcon("android.png");
  Image img = icon.getImage();
  for (int i=0; i<NUMANDROIDS; i++) {
    int x = (int)((i+0.25)*screenSize.width)/NUMANDROIDS;
    androidsArray[i] = new Android(img,x,50);
  }

  icon = new ImageIcon("player.png");
  img = icon.getImage();
  player = new Player(img,NUMANDROIDS/2,475);

  icon = new ImageIcon("bullet.png");
  img = icon.getImage();
  bullet = new Bullet(img,0,-100);

  // setup application window
  setBounds(50, 50, screenSize.width, screenSize.height);
  setVisible(true);
  setTitle("Android Attack!");

  // create and start animation thread
  Thread t = new Thread(this);
  t.start();

  // receive keyboard events
  addKeyListener(this);
}

// thread's entry point
public void run() {
  while ( 1==1 ) {
    // 1: sleep for 1/50 sec
    try {
      Thread.sleep(20);
    } catch (InterruptedException e) { }

    // 2: move game objects
    // TO DO


    // 3: force an application repaint
    this.repaint();
  }
}

// Three Keyboard Event-Handler functions
public void keyPressed(KeyEvent e) {
  // TO DO

}

public void keyReleased(KeyEvent e) {
  // TO DO

}

public void keyTyped(KeyEvent e) { }

public static Android getAndroidByIdx(int idx) {
  return androidsArray[idx];
  }

  // application's paint method
  public void paint(Graphics g) {
    // clear the canvas with a black rectangle
    g.setColor(Color.BLACK);
    g.fillRect(0, 0, screenSize.width, screenSize.height);

    // paint the game objects
    player.paint(g);
    bullet.paint(g);
    for ( int i=0; i<NUMANDROIDS; i++ )
      androidsArray[i].paint(g);
  }

  // application's entry point
  public static void main(String[] args) {
    AndroidAttack aa = new AndroidAttack();
  }
}
```

```
public class Android {
  // member data
  private int x, y;
  private Image image;

  // constructor
  public Android(Image i, int x, int y) {
    this.x = x;
    this.y = y;
    image = i;
  }

  public void move() {
    if (Math.random()<0.25)
      y++;
  }

  public int getX() {
    return x;
  }

  public int getY() {
    return y;
  }

  public void setY(int y) {
    this.y = y;
  }

  public void paint(Graphics g) {
    g.drawImage(image, x, y, null);
  }
}
```

```
public class Player {
  // member data
  // note that the player's horizontal (x) position is always locked to that of
  // a specific android, whose array index (in AndroidAttack.androidsArray)
  // is defined by androidIdx
  private int androidIdx, y, dx;
  private Image image;

  // constructor
```

```
   public Player(Image i, int androidIdx, int y) {
      this.androidIdx = androidIdx;
      this.y = y;
      image = i;
      dx = 0;
   }

   public void setMove(int keycode) {
      if (keycode==KeyEvent.VK_LEFT)
         dx = -1;
      else if (keycode==KeyEvent.VK_RIGHT)
         dx = 1;
   }

   public void unsetMove(int keycode) {
      if (keycode==KeyEvent.VK_LEFT || keycode==KeyEvent.VK_RIGHT)
         dx = 0;
   }

   public void move() {
      androidIdx += dx;
      if (androidIdx<0)
         androidIdx = 0;
      else if (androidIdx>=AndroidAttack.NUMANDROIDS)
         androidIdx = AndroidAttack.NUMANDROIDS-1;
   }

   public int getX() {
      Android a = AndroidAttack.getAndroidByIdx(androidIdx);
      return a.getX();
   }

   public int getY() {
      return y;
   }

   public int getAndroidIdx() {
      return androidIdx;
   }

   public void paint(Graphics g) {
      g.drawImage(image, getX(), y, null);
   }
}


public class Bullet {
   // member data
   // note that the bullet's horizontal (x) position is always locked to that of
   // a specific android, whose array index (in AndroidAttack.androidsArray)
   // is defined by androidIdx
   private int androidIdx, y;
   private Image image;

   // constructor
   public Bullet(Image i, int androidIdx, int y) {
      this.androidIdx = androidIdx;
      this.y = y;
      image = i;
   }

   public boolean move() {
      // returns true if the android we're locked onto should be killed and reset to
the top of the screen
      y -= 5;
      Android a = AndroidAttack.getAndroidByIdx(androidIdx);
      return (y<=a.getY());
    }

    public void SetPos(Player p) {
      this.androidIdx = p.getAndroidIdx();
      this.y = p.getY();
    }

    public int getX() {
      Android a = AndroidAttack.getAndroidByIdx(androidIdx);
      return a.getX();
    }

    public int getY() {
      return y;
    }

    public void setY(int y) {
      this.y = y;
    }

    public int getAndroidIdx() {
      return androidIdx;
    }

    public void paint(Graphics g) {
      g.drawImage(image, getX(), y, null);
    }
  }
```

PTO

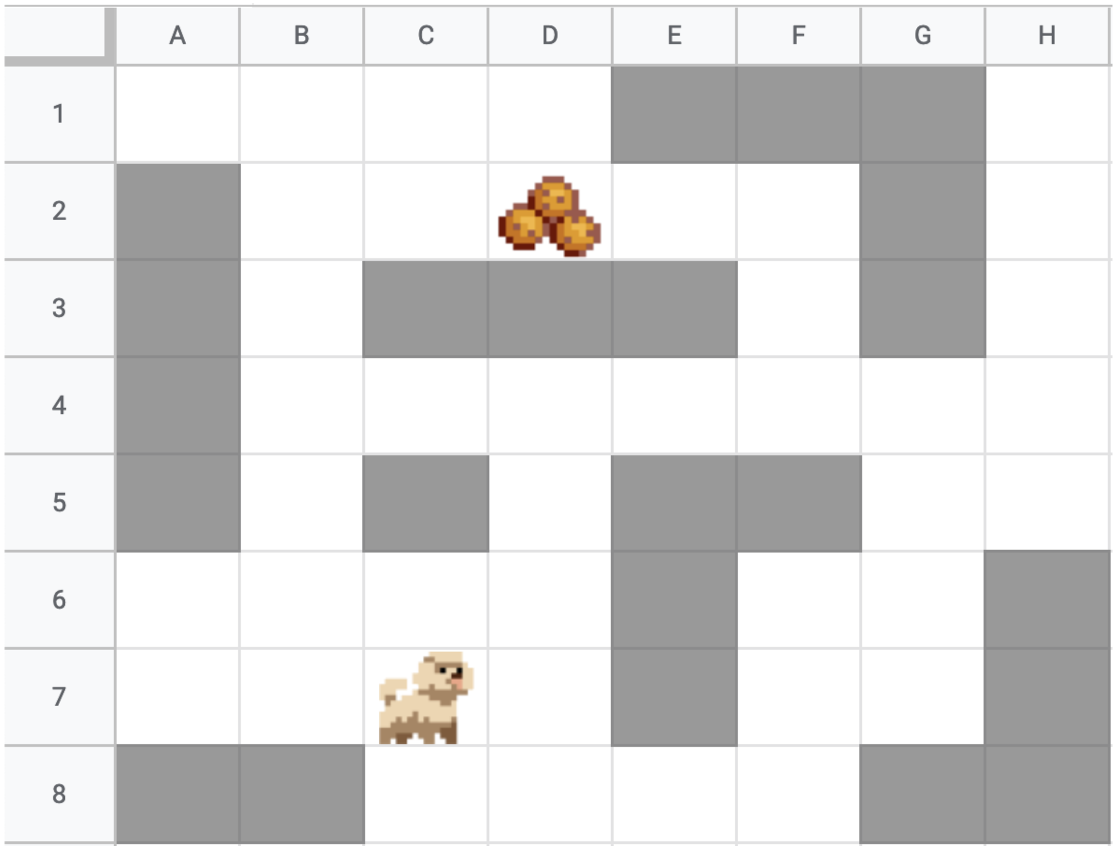
Q.4. A* Pathfinding

(i) With regard to the A* pathfinding algorithm, discuss the following terms: node, open list, parent node, expanding a node, f value, g value, h value

[5 marks]

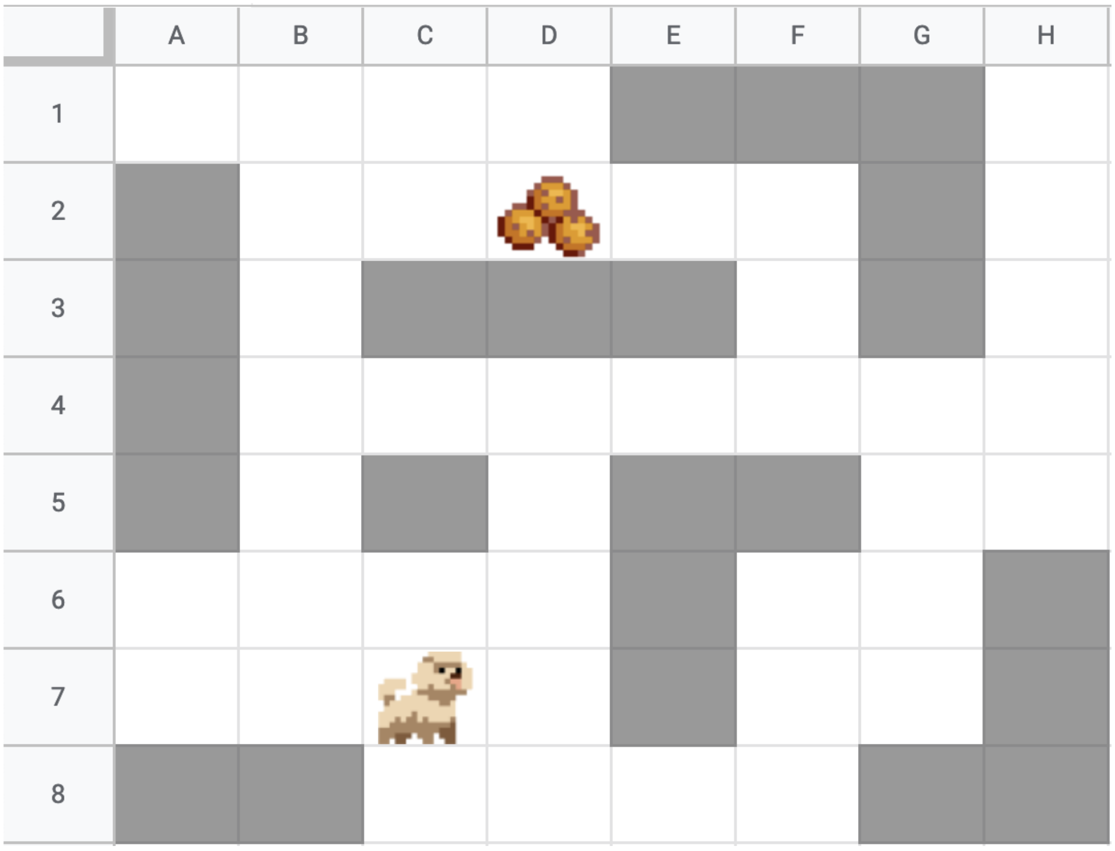
(ii) Joey the naughty dog is visiting Sam and needs to get through a maze to steal his cookies (see diagram below). Joey's movements may only be made horizontally or vertically (*not diagonally*), by a full square at a time. He will use A* pathfinding. The shaded squares represent impassable walls. Annotate and submit the diagram below (**detachable version on final page of this exam paper**) with F, G, H values and parent node indication arrows for each square that will have been considered (i.e. added to the open list) before the correct path is discovered.

[5 marks]



You may detach and submit this page as part of your answer to Q.4., if you wish

CT255
Student ID ______________________________


A
B
C
D
E
F
G
H
1
2
3
4
5
6
7
8