BTL 1

Câu 1: Cho trọng số của từng cấp bậc là 1, 2, 3, 4, 5, 7, 8, 9, 10, 12, 15, 18, 20, 30, 40, 50, 70, tổng sức mạnh lực lượng quân đội của 2 quân đoàn biết quân đoàn 1 và quân đoàn 2 có số lượng quân nhân lần lượt là [20, 0, 0, 10, 0, 0, 0, 10, 0, 0, 0, 2, 0, 0 , 0, 0, 0], [1, 1, 10, 1, 1, 0, 0 , 0, 0, 0, 10, 3, 0, 1, 1, 1, 1]

A. 622 B. 186

C. 436 D. 761

Câu 2: Nếu trọng số là 70 và số lượng thì là 200 thì sức mạnh lực lượng quân đội là bao nhiêu?

A. 14000 B. 1400

C. 270 D. 1000

Câu 3: Cho biết determineRightTarget mục đích chính là trả về tên tiếng Anh của mục tiêu chính cần đánh chiếm, nếu không tìm thấy ID hợp lệ, hàm trả về "INVALID". determineRightTarget(“P113IKU”) sẽ trả về cái gì?

A. KonTum B. INVALID

C. Pleiku D. National Route 14

Dành cho chuỗi câu 4-5:

Câu 4: Cho biết decodeTarget mục đích:

- Giải mã message bằng một trong hai phương pháp: Caesar Cipher hoặc Đảo ngược chuỗi.

– Sau khi giải mã, kiểm tra xem chuỗi thu được có khớp với Danh sách mục tiêu chính cần đánh chiếm không.

– Nếu có, trả về tên mục tiêu đã giải mã thành công. Nếu không, trả về "INVALID".

Quy tắc lựa chọn phương pháp giải mã:

• Nếu EXP1 ≥ 300 và EXP2 ≥ 300, thì sử dụng phương pháp Caesar Cipher.

• Nếu EXP1 < 300 hoặc EXP2 < 300, thì sử dụng phương pháp Đảo ngược chuỗi.

Phương pháp giải mã:

1. Caesar Cipher:

• Mỗi ký tự trong thông điệp bị dịch đi một số bước bằng (EXP1 + EXP2) mod 26 theo bảng chữ cái (trừ ký tự khoảng cách).

• Công thức dịch ký tự:

Ký tự mới = (Ký tự cũ − shift + 26) mod 26

• Nếu kết quả là một ký tự ngoài phạm vi chữ cái (A-Z, a-z), giữ nguyên ký tự đó.

2. Đảo ngược chuỗi: Đảo ngược toàn bộ chuỗi ký tự đầu vào.

Cho

```
int EXP1 = 596;
int EXP2 = 1000;
code = "NaTiOnAl RoUtE 14";
```

decodeTarget(code, EXP1, EXP2) có kết quả là

A. Invalid B. National Route 21

C. Pleiku D. National Route 14

Câu 5: Với

```
int EXP1 = -40;
int EXP2 = 2;
code = "PAl cUd";
string result = decodeTarget(code, EXP1, EXP2);
```

cho kết quả là

A. PAl cUd B. dUc lAP

C. Duc Lap D. INVALID

Câu 6: Điền vào dòng code sau biết

Tiếp tế sẽ được điều chỉnh theo công thức:

$$\Delta T_1 = \left(\frac{\mathbf{LF_1}}{\mathbf{LF_1} + \mathbf{LF_2}} \times (\mathbf{T_1} + \mathbf{T_2}) \right) \times \left(1 + \frac{\mathbf{EXP_1} - \mathbf{EXP_2}}{100} \right)$$

$$\Delta T_2 = (\mathbf{T_1} + \mathbf{T_2}) - \Delta T_1$$

$$\mathbf{T_1} = \mathbf{T_1} + \Delta T_1$$

$$\mathbf{T_2} = \mathbf{T_2} + \Delta T_2$$

Quy tắc điều chỉnh theo sự kiện lịch sử: Mã sự kiện lịch sử ảnh hưởng đến hậu cần được chia thành các khoảng giá trị sau:

- **Nếu** $\mathbf{E} = 0$: Không có sự kiện đặc biệt, áp dụng công thức trên.
- **Nếu** $\mathbf{E} \in [1, 9]$:
  - Quân đoàn 1 mất $(\mathbf{E} \times 1\%) \times \mathbf{T_1}$ tiếp tế.
  - Quân đoàn 2 mất $(\mathbf{E} \times 0.5\%) \times \mathbf{T_2}$ tiếp tế.
- **Nếu** $\mathbf{E} \in [10, 29]$:
  - Mỗi quân đoàn nhận thêm $(\mathbf{E} \times 50)$ đơn vị tiếp tế.
- **Nếu** $\mathbf{E} \in [30, 59]$:
  - Quân đoàn 1 được tăng cường $(\mathbf{E} \times 0.5\%) \times \mathbf{T_1}$ tiếp tế.
  - Quân đoàn 2 chỉ được tăng cường $(\mathbf{E} \times 0.2\%) \times \mathbf{T_2}$ tiếp tế.
- **Nếu** $\mathbf{E} \in [60, 99]$: Đường tiếp tế bị gián đoạn, tiếp tế không được điều chỉnh.

```
void manageLogistics(int LF1, int LF2, int EXP1, int EXP2, int &T1, int &T2, int E) {
 check_range(T1, MAX_T);
 check_range(T2, MAX_T);
 if (60 <= E && E <= 99) {
    return;
 }
 if (1 <= E && E <= 9) {
    // CODE 1 T1 = safeCeil(T1 * (1 - E * 0.01));
    // CODE 2 T2 = safeCeil(T2 * (1 - E * 0.005));
 } else if (10 <= E && E <= 29) {
    T1 = T1 + E * 50;
```

```
    T2 = T2 + E * 50;
  } else if (30 <= E && E <= 59) {
    T1 = safeCeil(T1 * (1 + E * 0.005));
    T2 = safeCeil(T2 * (1 + E * 0.002));
  } else {
    check_range(LF1, INT_MAX);
    check_range(LF2, INT_MAX);
    check_range(EXP1, MAX_EXP);
    check_range(EXP2, MAX_EXP);
    int totalSupply = T1 + T2;
    double deltaT1 = (float(LF1) / (LF1 + LF2) * totalSupply) * (1 + (EXP1 - EXP2) /
100.0);
    int dT1 = safeCeil(deltaT1);

    double deltaT2 = totalSupply - deltaT1;
    int dT2 = safeCeil(deltaT2);

    T1 += dT1;
    T2 += dT2;
  }
  check_range(T1, MAX_T);
  check_range(T2, MAX_T);
}
```

A. CODE 1: T1 = safeCeil(T1 * (1 - E * 0.01)); CODE 2: T2 = safeCeil(T2 * (1 - E * 0.005));

B. CODE 1: T1 = float (T1 * (1 - E * 0.01)); CODE 2: T2 = safeCeil(T2 * (1 - E * 0.005));

C. CODE 1: T1 = safeCeil (T1 * (1 - E * 0.01)); CODE 2: T2 = float(T2 * (1 - E * 0.005));

D. CODE 1: T1 = float (T1 * (1 - E * 0.01)); CODE 2: T2 = float (T2 * (1 - E * 0.005));

Từ câu 7 đến 10

Câu 7: Cho các dữ liệu sau:

LF1= 300, LF2=280, EXP1=450, EXP2=470, T1=2500, T2=2600 và ma trận BattleField như sau:

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
10	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**20**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**30**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**40**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**50**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**60**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**70**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**80**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**90**	**0**

Có quy trình xử lí như sau

- **Quy trình xử lý**:
  - Tính sức mạnh tổng hợp ban đầu:

$$S = (LF_1 + LF_2) + (EXP_1 + EXP_2) \times 5 + (T_1 + T_2) \times 2$$

  - Duyệt ma trận và cập nhật S tại mỗi vị trí (i, j):
    - Hàng lẻ:

$$S = S - \left(\frac{\mathbf{battleField}_{i,j} \times 3}{2} \right)$$

    - Hàng chẵn:

$$S = S - \left(\frac{\mathbf{battleField}_{i,j} \times 2}{3} \right)$$

Giá trị sức mạnh tổng hợp S là:

A. 15380 **B. 14872**

C. 14871 D. 15381

Câu 8: Đoạn mã nào dùng để tính S sau khi duyệt từng phần tử

A.

```
for (int i = 0; i < 10; i++) {
   for (int j = 0; j < 10; j++) {
     if (i % 2 == 0) {
       S -= battleField[i][j] * 2.0 / 3;
     } else {
       S -= battleField[i][j] * 3.0 / 2;
     }
   }
 }
```

B.

```
for (int i = 1; i <= 10; i++) {
    for (int j = 1; j <= 10; j++) {
      if (i % 2 == 0) {
        S -= battleField[i][j] * 2.0 / 3;
      } else {
        S -= battleField[i][j] * 3.0 / 2;
      }
    }
  }
```

C.

```
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
      if (i % 2 == 1) {
        S -= battleField[i][j] * 2.0 / 3;
      } else {
        S -= battleField[i][j] * 3.0 / 2;
      }
```

Câu 9: Vấn đề gì sẽ xảy ra nếu chúng ta duyệt từ 0 tới 10 cho I và j, và khai báo int battleField[11][11]; và không truyền giá trị của từng phần tử vào cho những mảng khai báo dư

Chương trình sẽ luôn chạy đúng vì C++ tự động khởi tạo mảng về 0

B. Chương trình sẽ báo lỗi biên dịch do chưa khởi tạo mảng

C. Chương trình vẫn chạy nhưng có thể cho kết quả sai hoặc không xác định do dùng giá trị rác trong mảng

D. Trình biên dịch sẽ tự động gán giá trị ngẫu nhiên cho phần tử mảng

Câu 10: Điều kiện nào sau đây sẽ khiến hàm planAttack(...) trả về số âm?

A. battleField có giá trị tổng nhỏ

B. LF1, LF2, EXP1, EXP2 đều bằng 0

C. Các giá trị trong battleField đủ lớn để khiến S < 0

D. Tất cả đều đúng

BTL 2

Chuỗi câu 11-13:

Trong BTL2 cho hàm class Position như sau:

```
class Position
{
private:
    int r, c;

public:
    Position(int r = 0, int c = 0);
    Position(const string &str_pos);
    string str() const;
    // Các method bổ sung đã được ẩn đi
};
```

Câu 11: Hàm khởi tạo Position(int r = 0, int c = 0) có ý nghĩa gì?

A. Chỉ cho phép khởi tạo đối tượng khi có đầy đủ tham số
B. Cho phép khởi tạo đối tượng với giá trị mặc định nếu không truyền tham số
C. Không phải là hàm khởi tạo mặc định
D. Bắt buộc phải truyền giá trị r và c khi tạo đối tượng

Câu 12: Từ khóa const sau các hàm như int getRow() const; có tác dụng gì?

A. Ngăn không cho sử dụng hàm trong các đối tượng hằng
B. Không có tác dụng gì
C. Chỉ ra rằng hàm không thay đổi trạng thái của đối tượng
D. Cho phép hàm thay đổi các biến thành viên

Câu 13: Để thay đổi giá trị của r và c trong một đối tượng Position, ta nên dùng cách nào?

A. Gọi trực tiếp p.r = 5; p.c = 7;
B. Gọi hàm setRow() và setCol()
C. Khởi tạo lại đối tượng
D. Gọi hàm getRow() và getCol()

Chuỗi câu 14-15:

Câu 14: Cho đoạn code sau:

```
class Unit
{
protected:
   int quantity, weight;
   Position pos;
   int attackScore;
public:
   int AttackScore() const {return attackScore;}
   virtual int getType() const = 0;
   // Đã ẩn các method khác
};

class Vehicle : public Unit
{
private:
   VehicleType vehicleType;

public:
   // Đã ẩn các method khác
};

class Infantry : public Unit
{
private:
   InfantryType infantryType;

public:
        // Đã ẩn các method khác
};
```

Phát biểu nào sau đây là đúng:

A. Hàm AttackScore() chỉ có thể được gọi từ đối tượng kiểu Unit, không thể gọi từ Vehicle hay Infantry.
B. Vì AttackScore() không là virtual, nên khi gọi từ con trỏ kiểu Unit* trỏ đến Vehicle hoặc Infantry, kết quả có thể không chính xác.
C. Hàm AttackScore() có thể được gọi từ mọi đối tượng của các lớp kế thừa như Vehicle và Infantry vì nó là phương thức công khai trong lớp cơ sở.
D. Hàm AttackScore() không thể được truy cập bên ngoài lớp Unit vì attackScore là thuộc tính protected.

Câu 15: Điều gì sẽ xảy ra nếu bạn không override phương thức getType() trong cả Vehicle và Infantry?

A. Không có vấn đề gì, chương trình vẫn biên dịch bình thường.

B. Chương trình chỉ không biên dịch nếu bạn gọi getType() từ đối tượng Vehicle hoặc Infantry.

C. Chương trình chỉ biên dịch nếu Vehicle và Infantry là lớp ảo.

D. Chương trình không biên dịch, vì Vehicle và Infantry vẫn là lớp trừu tượng nếu không override getType().

Chuỗi câu 16-18:

```
bool UnitList::insert(Unit* unit)
{
   if (unit == NULL) return false;

   Node* newNode = new Node(unit);

   bool isVehicle = dynamic_cast<Vehicle*>(unit);
   bool isInfantry = dynamic_cast<Infantry*>(unit);

   if (!isVehicle && !isInfantry)
   {
      delete newNode;
      return false;
   }

   if (!head && !tail)
   {
      // CODE 1
      return true;
   }

   Node* current = head;
```

```
    while (current)
    {
        if ((isVehicle && dynamic_cast<Vehicle*>(current->unit) && current->unit->getType() == unit->getType()) ||
            (isInfantry && dynamic_cast<Infantry*>(current->unit) && current->unit->getType() == unit->getType()))
        {
            // CODE 2
            delete newNode;
            return false;
        }
        current = current->next;
    }

    // CODE 3

    return true;
}
```

Câu 16: CODE 1:

A. head = tail = newNode; isVehicle ? ++countVehicle : ++countInfantry;

B. countVehicle = countInfantry = 1;

C. tail = head = newNode; ++countVehicle;

D. head = newNode; tail = NULL;

Câu 17: CODE 2:

A. current->unit->add(unit);

B. current->unit->setQuantity(current->unit->getQuantity() + unit->getQuantity()); current->unit->setWeight(fmax(current->unit->getWeight(), unit->getWeight()));

C. current->unit->merge(unit);

D. unit->setQuantity(0);

Câu 18: CODE 3:

A. if (isVehicle) { tail->next = newNode; tail = newNode; ++countVehicle; } else { newNode->next = head; head = newNode; ++countInfantry; }

B. tail = newNode; ++countVehicle;

C. head = newNode; ++countInfantry;

D. return false;

Câu 19: Cho đoạn code sau

```
bool UnitList::isContain(InfantryType infantryType)
{
   Node* current = head;
   // CODE
   {
      if (current->unit->getType() == infantryType)
      {
         if (dynamic_cast<Infantry*>(current->unit))
         {
            return true;
         }
      }
      current = current->next;
   }
   return false; }
```

CODE sẽ là ?

A. for (; current != nullptr;)
B. for (Node* current = head ; current != nullptr; current = current->next;)
C. for (; current != nullptr; current = current->next;)
D. for (current != nullptr)

Câu 20: Cho code sau

```
UnitList::~UnitList()
{
   Node* current = head;
   while (current != nullptr)
   {
      Node* next = current->next;
      delete current->next;    //(1)
      delete current;    //(2)
      current = next;   //(3)
   }
}
```

Code trên chạy lỗi ở đâu ?

A. Ở dòng (1) vì truy cập current->next khi current có thể là nullptr
B. Ở dòng 2 – Vì đang delete current->next, nhưng sau đó vẫn sử dụng lại next trỏ vào vùng nhớ đã bị xóa
C. Ở dòng 3 – Vì delete current ngay sau khi đã delete current->next gây double-delete
D. Không có lỗi

Câu 21: Giả sử có mảng 2 chiều terrain kiểu TerrainElement* đã được cấp phát và khởi tạo bằng nullptr. Để gán một đối tượng Mountain vào vị trí (row, col) dựa trên Position* pos. Điều kiện nào dưới đây là đúng và đầy đủ nhất để đảm bảo việc gán là an toàn và hợp lệ?**

A. if (terrain[pos->getRow()][pos->getCol()] == nullptr)

B. if (pos->getRow() >= 0 && pos->getCol() >= 0)
C. if (pos->getRow() >= 0 && pos->getRow() < n_rows &&
pos->getCol() >= 0 && pos->getCol() < n_cols &&
terrain[pos->getRow()][pos->getCol()] == nullptr)
D. if (terrain != nullptr && pos != nullptr)

Câu 22: Cho đoạn code dưới đây

```
UnitList* list = new UnitList(100);
Vehicle* tank1 = new Vehicle(2, 5, Position(1, 1), TANK);
Vehicle* tank2 = new Vehicle(3, 3, Position(2, 2), TANK);
list->insert(tank1);
list->insert(tank2);
list->str() : sẽ trả về
```

A.
UnitList[count_vehicle=2;count_infantry=0;Vehicle[vehicleType=TANK,quantity=2,weight=5,position=(1,1)],Vehicle[vehicleType=TANK,quantity=3,weight=3,position=(2,2)]]

B.
UnitList[count_vehicle=1;count_infantry=0;Vehicle[vehicleType=TANK,quantity=3,weight=3,position=(2,2)]]

C.
UnitList[count_vehicle=1;count_infantry=0;Vehicle[vehicleType=TANK,quantity=5,weight=5,position=(1,1)]]

D.
UnitList[count_vehicle=1;count_infantry=0;Vehicle[vehicleType=TANK,quantity=5,weight=8,position=(1,1)]]