----------Practice Questions----------

1.alloy composition

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int sampleID;
    char name[50];
    union {
        float metalA;
        float metalB;
        float metalC;
    } composition;
} Alloy;

void displayAlloyDetails(const Alloy* alloy) {
    printf("Sample ID: %d\n", alloy->sampleID);
    printf("Name: %s\n", alloy->name);
    printf("Composition: Metal A = %.2f%%, Metal B = %.2f%%, Metal C = %.2f%%\n",
           alloy->composition.metalA, alloy->composition.metalB, alloy->composition.metalC);
}

int main() {
    Alloy* alloys;
    int n;

    printf("Enter number of alloy samples: ");
    scanf("%d", &n);
```

```
    alloys = (Alloy*)malloc(n * sizeof(Alloy));

    for (int i = 0; i < n; i++) {
        printf("Enter details for sample %d:\n", i + 1);
        printf("Sample ID: ");
        scanf("%d", &alloys[i].sampleID);
        printf("Name: ");
        scanf("%s", alloys[i].name);
        printf("Enter Metal A composition percentage: ");
        scanf("%f", &alloys[i].composition.metalA);
        printf("Enter Metal B composition percentage: ");
        scanf("%f", &alloys[i].composition.metalB);
        printf("Enter Metal C composition percentage: ");
        scanf("%f", &alloys[i].composition.metalC);
    }

    for (int i = 0; i < n; i++) {
        displayAlloyDetails(&alloys[i]);
    }

    free(alloys);
    return 0;
}
```

2.heat treatment

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
typedef struct {
    int processID;
    float temperature;
    float duration;
    float coolingRate;
} HeatTreatmentProcess;

void displayProcessDetails(const HeatTreatmentProcess* process) {
    printf("Process ID: %d\n", process->processID);
    printf("Temperature: %.2f\n", process->temperature);
    printf("Duration: %.2f\n", process->duration);
    printf("Cooling Rate: %.2f\n", process->coolingRate);
}

int main() {
    HeatTreatmentProcess* processes;
    int n;

    printf("Enter number of heat treatment processes: ");
    scanf("%d", &n);

    processes = (HeatTreatmentProcess*)malloc(n * sizeof(HeatTreatmentProcess));

    for (int i = 0; i < n; i++) {
        printf("Enter details for process %d:\n", i + 1);
        printf("Process ID: ");
        scanf("%d", &processes[i].processID);
        printf("Temperature: ");
```

```
        scanf("%f", &processes[i].temperature);
        printf("Duration: ");
        scanf("%f", &processes[i].duration);
        printf("Cooling Rate: ");
        scanf("%f", &processes[i].coolingRate);
    }

    for (int i = 0; i < n; i++) {
        displayProcessDetails(&processes[i]);
    }

    free(processes);
    return 0;
}
```

3.steel quality

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int testID;
    char type[50];
    union {
        float tensileStrength;
        float hardness;
        float elongation;
    } result;
} SteelTest;
```

```
void displayTestDetails(const SteelTest* test) {
    printf("Test ID: %d\n", test->testID);
    printf("Test Type: %s\n", test->type);
    printf("Result (Tensile Strength): %.2f\n", test->result.tensileStrength);
}

int main() {
    SteelTest* tests;
    int n;

    printf("Enter number of steel tests: ");
    scanf("%d", &n);

    tests = (SteelTest*)malloc(n * sizeof(SteelTest));

    for (int i = 0; i < n; i++) {
        printf("Enter details for test %d:\n", i + 1);
        printf("Test ID: ");
        scanf("%d", &tests[i].testID);
        printf("Test Type: ");
        scanf("%s", tests[i].type);
        printf("Enter Tensile Strength: ");
        scanf("%f", &tests[i].result.tensileStrength);
    }

    for (int i = 0; i < n; i++) {
        displayTestDetails(&tests[i]);
    }
```

```
    free(tests);
    return 0;
}
```

4.metal fatigue

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    int materialID;
    char name[50];
    float enduranceLimit;
} MetalMaterial;

void displayMaterialDetails(const MetalMaterial* material) {
    printf("Material ID: %d\n", material->materialID);
    printf("Material Name: %s\n", material->name);
    printf("Endurance Limit: %.2f\n", material->enduranceLimit);
}

int main() {
    MetalMaterial* materials;
    int n;

    printf("Enter number of materials: ");
    scanf("%d", &n);

    materials = (MetalMaterial*)malloc(n * sizeof(MetalMaterial));
```

```
    for (int i = 0; i < n; i++) {
        printf("Enter details for material %d:\n", i + 1);
        printf("Material ID: ");
        scanf("%d", &materials[i].materialID);
        printf("Material Name: ");
        scanf("%s", materials[i].name);
        printf("Enter Endurance Limit: ");
        scanf("%f", &materials[i].enduranceLimit);
    }

    for (int i = 0; i < n; i++) {
        displayMaterialDetails(&materials[i]);
    }

    free(materials);
    return 0;
}
```

5.foundary management system

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int castingID;
    float weight;
    char material[50];
    union {
        float dimensions[3];
```

```
        float thermalConductivity;
    } moldProperties;
} Casting;

void displayCastingDetails(const Casting* casting) {
    printf("Casting ID: %d\n", casting->castingID);
    printf("Weight: %.2f\n", casting->weight);
    printf("Material: %s\n", casting->material);
    printf("Mold Dimensions: %.2f x %.2f x %.2f\n", casting->moldProperties.dimensions[0], casting->moldProperties.dimensions[1], casting->moldProperties.dimensions[2]);
}

int main() {
    Casting* castings;
    int n;

    printf("Enter number of castings: ");
    scanf("%d", &n);

    castings = (Casting*)malloc(n * sizeof(Casting));

    for (int i = 0; i < n; i++) {
        printf("Enter details for casting %d:\n", i + 1);
        printf("Casting ID: ");
        scanf("%d", &castings[i].castingID);
        printf("Weight: ");
        scanf("%f", &castings[i].weight);
        printf("Material: ");
        scanf("%s", castings[i].material);
```

```
        printf("Enter Mold Dimensions (L x W x H): ");
        scanf("%f %f %f", &castings[i].moldProperties.dimensions[0],
&castings[i].moldProperties.dimensions[1], &castings[i].moldProperties.dimensions[2]);
    }

    for (int i = 0; i < n; i++) {
        displayCastingDetails(&castings[i]);
    }

    free(castings);
    return 0;
}
```

6.

metal purity analysis

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int sampleID;
    char type[50];
    float purity;
    union {
        float traceElements;
        float oxides;
    } impurities;
} MetalSample;

void displaySampleDetails(const MetalSample* sample) {
```

```
    printf("Sample ID: %d\n", sample->sampleID);
    printf("Sample Type: %s\n", sample->type);
    printf("Purity: %.2f%%\n", sample->purity);
}

int main() {
    MetalSample* samples;
    int n;

    printf("Enter number of samples: ");
    scanf("%d", &n);

    samples = (MetalSample*)malloc(n * sizeof(MetalSample));

    for (int i = 0; i < n; i++) {
        printf("Enter details for sample %d:\n", i + 1);
        printf("Sample ID: ");
        scanf("%d", &samples[i].sampleID);
        printf("Sample Type: ");
        scanf("%s", samples[i].type);
        printf("Enter Purity: ");
        scanf("%f", &samples[i].purity);
    }

    for (int i = 0; i < n; i++) {
        displaySampleDetails(&samples[i]);
    }

    free(samples);
```

```
    return 0;
}
```

7.

corrosion testing system

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    int testID;
    float duration;
    char environment[100];
} CorrosionTest;

void displayTestDetails(const CorrosionTest* test) {
    printf("Test ID: %d\n", test->testID);
    printf("Duration: %.2f hours\n", test->duration);
    printf("Environment: %s\n", test->environment);
}

int main() {
    CorrosionTest* tests;
    int n;

    printf("Enter number of corrosion tests: ");
    scanf("%d", &n);

    tests = (CorrosionTest*)malloc(n * sizeof(CorrosionTest));
```

```
    for (int i = 0; i < n; i++) {
        printf("Enter details for test %d:\n", i + 1);
        printf("Test ID: ");
        scanf("%d", &tests[i].testID);
        printf("Duration (in hours): ");
        scanf("%f", &tests[i].duration);
        printf("Environment (e.g., saline, atmospheric): ");
        scanf("%s", tests[i].environment);
    }

    for (int i = 0; i < n; i++) {
        displayTestDetails(&tests[i]);
    }

    free(tests);
    return 0;
}
```

8.

welding parameter optimisation

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int paramID;
    float voltage;
    float current;
    float speed;
```

```
} WeldingParameter;

typedef union {
    char weldingType[10];  // MIG, TIG, Arc
} WeldingType;

void displayWeldingDetails(const WeldingParameter* param, const WeldingType* type) {
    printf("Parameter ID: %d\n", param->paramID);
    printf("Voltage: %.2f V\n", param->voltage);
    printf("Current: %.2f A\n", param->current);
    printf("Speed: %.2f m/min\n", param->speed);
    printf("Welding Type: %s\n", type->weldingType);
}

int main() {
    WeldingParameter* parameters;
    WeldingType* types;
    int n;

    printf("Enter number of welding parameters: ");
    scanf("%d", &n);

    parameters = (WeldingParameter*)malloc(n * sizeof(WeldingParameter));
    types = (WeldingType*)malloc(n * sizeof(WeldingType));

    for (int i = 0; i < n; i++) {
        printf("Enter details for welding parameter set %d:\n", i + 1);
        printf("Parameter ID: ");
        scanf("%d", &parameters[i].paramID);
```

```
        printf("Voltage: ");
        scanf("%f", &parameters[i].voltage);
        printf("Current: ");
        scanf("%f", &parameters[i].current);
        printf("Speed: ");
        scanf("%f", &parameters[i].speed);
        printf("Enter Welding Type (MIG, TIG, Arc): ");
        scanf("%s", types[i].weldingType);
    }

    for (int i = 0; i < n; i++) {
        displayWeldingDetails(&parameters[i], &types[i]);
    }

    free(parameters);
    free(types);
    return 0;
}
```

9.metal surface finish analysys

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    int configID;
    char material[30];
    char units[10];
} Config;
```

```
void analyzeSurface(double **measurements, int size, Config *const config) {
    printf("Configuration ID: %d\nMaterial: %s\nUnits: %s\n", config->configID, config->material, config->units);
    printf("Surface measurements:\n");
    for (int i = 0; i < size; i++) {
        printf("%.2lf ", (*measurements)[i]);
    }
    printf("\n");
}

int main() {
    Config config = {1, "Steel", "Microns"};
    int size = 5;
    double *measurements = (double *)malloc(size * sizeof(double));
    if (!measurements) return -1;

    printf("Enter %d surface measurements: ", size);
    for (int i = 0; i < size; i++) {
        scanf("%lf", &measurements[i]);
    }

    analyzeSurface(&measurements, size, &config);

    free(measurements);
    return 0;
}
```

10.

smelting process tracker

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    int processID;
    char oreType[20];
    double temperature;
} Process;

typedef union {
    char quality[20];
    double impurityLevel;
} OreProperties;

void trackSmelting(Process *const process, OreProperties *oreProp, double **heatData, int size) {
    printf("Process ID: %d\nOre Type: %s\nTemperature: %.2lf\n", process->processID, process->oreType,
process->temperature);
    printf("Heat data:\n");
    for (int i = 0; i < size; i++) {
        printf("%.2lf ", (*heatData)[i]);
    }
    printf("\n");
}

int main() {
    Process process = {101, "Iron Ore", 1500.5};
    OreProperties oreProp = {.quality = "High"};
```

```
    int size = 4;
    double *heatData = (double *)malloc(size * sizeof(double));
    if (!heatData) return -1;

    printf("Enter %d heat data points: ", size);
    for (int i = 0; i < size; i++) {
        scanf("%lf", &heatData[i]);
    }

    trackSmelting(&process, &oreProp, &heatData, size);

    free(heatData);
    return 0;
}
```

11.

electroplating system simulation

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    char ionType[20];
    int charge;
    double concentration;
} Ion;

void simulatePlating(Ion *const ion, double **params, int size, char *electrolyte) {
```

```
    printf("Ion Type: %s\nCharge: %d\nConcentration: %.2lf\nElectrolyte: %s\n", ion->ionType, ion->charge, ion->concentration, electrolyte);
    printf("Plating parameters:\n");
    for (int i = 0; i < size; i++) {
        printf("%.2lf ", (*params)[i]);
    }
    printf("\n");
}

int main() {
    Ion ion = {"Copper", 2, 0.05};
    char electrolyte[20] = "Sulfuric Acid";

    int size = 3;
    double *params = (double *)malloc(size * sizeof(double));
    if (!params) return -1;

    printf("Enter %d plating parameters: ", size);
    for (int i = 0; i < size; i++) {
        scanf("%lf", &params[i]);
    }

    simulatePlating(&ion, &params, size, electrolyte);

    free(params);
    return 0;
}
```

12.

casting defect analysis

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    int castingID;
    char material[20];
    double dimensions[3];
} Casting;

typedef union {
    char defectType[20];
    double severity;
} Defect;

void analyzeDefects(Casting *const casting, Defect *defect, double **data, int size) {
    printf("Casting ID: %d\nMaterial: %s\nDimensions: %.2lf x %.2lf x %.2lf\n", casting->castingID, casting->material, casting->dimensions[0], casting->dimensions[1], casting->dimensions[2]);
    printf("Defect data:\n");
    for (int i = 0; i < size; i++) {
        printf("%.2lf ", (*data)[i]);
    }
    printf("\n");
}

int main() {
    Casting casting = {301, "Aluminum", {10.5, 15.2, 7.8}};
    Defect defect = {.defectType = "Porosity"};
```

```
    int size = 4;
    double *data = (double *)malloc(size * sizeof(double));
    if (!data) return -1;

    printf("Enter %d defect data points: ", size);
    for (int i = 0; i < size; i++) {
        scanf("%lf", &data[i]);
    }

    analyzeDefects(&casting, &defect, &data, size);

    free(data);
    return 0;
}
```

13.

mettelugical lab automation

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    int sampleID;
    char sampleType[20];
    double dimensions[3];
} Sample;

void automateLab(Sample *const sample, double **testResults, int size, char *equipment) {
```

```
    printf("Sample ID: %d\nSample Type: %s\nDimensions: %.2lf x %.2lf x %.2lf\nEquipment: %s\n",
        sample->sampleID, sample->sampleType, sample->dimensions[0], sample->dimensions[1], sample->dimensions[2], equipment);
    printf("Test results:\n");
    for (int i = 0; i < size; i++) {
        printf("%.2lf ", (*testResults)[i]);
    }
    printf("\n");
}

int main() {
    Sample sample = {501, "Steel", {10.0, 5.5, 2.0}};
    char equipment[20] = "Microscope";

    int size = 3;
    double *testResults = (double *)malloc(size * sizeof(double));
    if (!testResults) return -1;

    printf("Enter %d test results: ", size);
    for (int i = 0; i < size; i++) {
        scanf("%lf", &testResults[i]);
    }

    automateLab(&sample, &testResults, size, equipment);

    free(testResults);
    return 0;
}
```

14.

metal hardness testing system

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    int testID;
    char method[20];
    double result;
} HardnessTest;

typedef union {
    char scale[10];
    double value;
} HardnessScale;

void testHardness(HardnessTest *const test, double **hardnessValues, int size, HardnessScale *scale) {
    printf("Test ID: %d\nMethod: %s\nResult: %.2lf\nScale: %s\n",
        test->testID, test->method, test->result, scale->scale);
    printf("Hardness values:\n");
    for (int i = 0; i < size; i++) {
        printf("%.2lf ", (*hardnessValues)[i]);
    }
    printf("\n");
}

int main() {
    HardnessTest test = {601, "Brinell", 450.0};
```

```
    HardnessScale scale = {.scale = "HB"};

    int size = 4;
    double *hardnessValues = (double *)malloc(size * sizeof(double));
    if (!hardnessValues) return -1;

    printf("Enter %d hardness values: ", size);
    for (int i = 0; i < size; i++) {
        scanf("%lf", &hardnessValues[i]);
    }

    testHardness(&test, &hardnessValues, size, &scale);

    free(hardnessValues);
    return 0;
}
```

15.

powder mettlurgy process tracker

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    int materialID;
    char materialType[20];
    double density;
} Material;
```

```
typedef union {
    char property[20];
    double percentage;
} PowderProperties;

void trackPowder(Material *const material, double **particleSizes, int size, PowderProperties
*properties) {
    printf("Material ID: %d\nMaterial Type: %s\nDensity: %.2lf\nPowder Property: %s\n",
        material->materialID, material->materialType, material->density, properties->property);
    printf("Particle sizes:\n");
    for (int i = 0; i < size; i++) {
        printf("%.2lf ", (*particleSizes)[i]);
    }
    printf("\n");
}

int main() {
    Material material = {701, "Aluminum", 2.7};
    PowderProperties properties = {.property = "Fine Grain"};

    int size = 5;
    double *particleSizes = (double *)malloc(size * sizeof(double));
    if (!particleSizes) return -1;

    printf("Enter %d particle sizes: ", size);
    for (int i = 0; i < size; i++) {
        scanf("%lf", &particleSizes[i]);
    }
```

```
    trackPowder(&material, &particleSizes, size, &properties);

    free(particleSizes);
    return 0;
}
```

16.

metal recycling amalysys

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    int materialID;
    char materialType[20];
    char recyclingMethod[30];
} RecyclingData;

void analyzeRecycling(RecyclingData *const data, double **impurityLevels, int size) {
    printf("Material ID: %d\nMaterial Type: %s\nRecycling Method: %s\n",
        data->materialID, data->materialType, data->recyclingMethod);
    printf("Impurity levels:\n");
    for (int i = 0; i < size; i++) {
        printf("%.2lf ", (*impurityLevels)[i]);
    }
    printf("\n");
}

int main() {
```

```
    RecyclingData data = {801, "Copper", "Electrolysis"};

    int size = 3;
    double *impurityLevels = (double *)malloc(size * sizeof(double));
    if (!impurityLevels) return -1;

    printf("Enter %d impurity levels: ", size);
    for (int i = 0; i < size; i++) {
        scanf("%lf", &impurityLevels[i]);
    }

    analyzeRecycling(&data, &impurityLevels, size);

    free(impurityLevels);
    return 0;
}
```

17.

rolling mill performance tracker

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    int millID;
    double rollDiameter;
    double speed;
} Mill;
```

```
void trackPerformance(Mill *const mill, double **outputData, int size, char *materialType) {
    printf("Mill ID: %d\nRoll Diameter: %.2lf\nSpeed: %.2lf\nMaterial Type: %s\n",
        mill->millID, mill->rollDiameter, mill->speed, materialType);
    printf("Output data:\n");
    for (int i = 0; i < size; i++) {
        printf("%.2lf ", (*outputData)[i]);
    }
    printf("\n");
}

int main() {
    Mill mill = {901, 1.5, 50.0};
    char materialType[20] = "Steel";

    int size = 4;
    double *outputData = (double *)malloc(size * sizeof(double));
    if (!outputData) return -1;

    printf("Enter %d output data points: ", size);
    for (int i = 0; i < size; i++) {
        scanf("%lf", &outputData[i]);
    }

    trackPerformance(&mill, &outputData, size, materialType);

    free(outputData);
    return 0;
}
```

18.

Thermal expansion analysis

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    int materialID;
    char materialType[20];
    double coefficient;
} MaterialExpansion;

typedef union {
    double thermalCoefficient;
    char range[20];
} ExpansionProperties;

void analyzeExpansion(MaterialExpansion *const material, double **tempData, int size,
ExpansionProperties *properties) {
    printf("Material ID: %d\nMaterial Type: %s\nExpansion Coefficient: %.2lf\nRange: %s\n",
        material->materialID, material->materialType, material->coefficient, properties->range);
    printf("Temperature data:\n");
    for (int i = 0; i < size; i++) {
        printf("%.2lf ", (*tempData)[i]);
    }
    printf("\n");
}

int main() {
```

```
    MaterialExpansion material = {1001, "Aluminum", 0.00023};
    ExpansionProperties properties = {.range = "High Temp"};

    int size = 3;
    double *tempData = (double *)malloc(size * sizeof(double));
    if (!tempData) return -1;

    printf("Enter %d temperature data points: ", size);
    for (int i = 0; i < size; i++) {
        scanf("%lf", &tempData[i]);
    }

    analyzeExpansion(&material, &tempData, size, &properties);

    free(tempData);
    return 0;
}
```

19.

metal melting point analyzer

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    int metalID;
    char metalName[30];
    double meltingPoint;
} MetalDetails;
```

```
void analyzeMeltingPoint(const MetalDetails *metal, double **temperatureData, int size) {
    printf("Metal ID: %d\nMetal Name: %s\nMelting Point: %.2lf°C\n",
           metal->metalID, metal->metalName, metal->meltingPoint);
    printf("Recorded Temperature Data:\n");
    for (int i = 0; i < size; i++) {
        printf("%.2lf ", (*temperatureData)[i]);
    }
    printf("\n");
}

int main() {
    MetalDetails metal = {1001, "Aluminum", 660.32};

    int size;
    printf("Enter the number of temperature records: ");
    scanf("%d", &size);

    double *temperatureData = (double *)malloc(size * sizeof(double));
    if (!temperatureData) {
        printf("Memory allocation failed!\n");
        return -1;
    }

    printf("Enter %d temperature data points: ", size);
    for (int i = 0; i < size; i++) {
        scanf("%lf", &temperatureData[i]);
    }
```

```
    analyzeMeltingPoint(&metal, &temperatureData, size);

    free(temperatureData);
    return 0;
}
```

20.

smelting effeciency analyzer

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    int processID;
    char oreType[30];
    double efficiency;
} ProcessDetails;

typedef union {
    double energyConsumption;
    double processDuration;
} ProcessParameters;

void analyzeSmeltingEfficiency(const ProcessDetails *process, double **energyData, int size,
ProcessParameters *parameters) {
    printf("Process ID: %d\nOre Type: %s\nEfficiency: %.2lf%%\n",
        process->processID, process->oreType, process->efficiency);
    printf("Energy Consumption Data:\n");
    for (int i = 0; i < size; i++) {
```

```
        printf("%.2lf ", (*energyData)[i]);
    }
    printf("\n");

    printf("Variable Process Parameter:\n");
    printf("Energy Consumption: %.2lf kWh\n", parameters->energyConsumption);
}

int main() {
    ProcessDetails process = {2001, "Iron Ore", 85.5};
    ProcessParameters parameters = {.energyConsumption = 1200.0};

    int size;
    printf("Enter the number of energy consumption records: ");
    scanf("%d", &size);

    double *energyData = (double *)malloc(size * sizeof(double));
    if (!energyData) {
        printf("Memory allocation failed!\n");
        return -1;
    }

    printf("Enter %d energy consumption data points (in kWh): ", size);
    for (int i = 0; i < size; i++) {
        scanf("%lf", &energyData[i]);
    }

    analyzeSmeltingEfficiency(&process, &energyData, size, &parameters);
```

```
    free(energyData);
    return 0;
}
```