Assignments:

Part1:

1) Swap two integers. By address & by ref show difference

2) Input and Output from Array using pointers (Raw pointers).

3) Line Editor

4) Employee with Dynamic Allocation and Highlight Menu (allow the user to specify the size of array at runtime) using Modern Pointers.

5) Continue Line Editor and make it with dynamic allocation(using Raw Pointers)

6) (self Study) pointer to pointer(using Raw ,using Modern).(not given at lecture)

```cpp
#include <iostream>
using namespace std;
int sum(int n1,int n2){
    return n1+n2;
}
int sum(int n1,int n2,int n3){
    return n1+n2+n3;
}
int main()
{
    cout<<sum(1,2)<<endl;
    cout<<sum(1,2,3)<<endl;
    return 0;
}
```

```cpp
// Swap by address
void swapByAddress(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Swap by reference
void swapByReference(int& a, int& b) {
    int temp = a;
    a = b;
    b = temp;
}

int main() {
    int x = 10, y = 20;
    swapByAddress(&x, &y);
    cout << "After swap by address: x=" << x << " y=" << y << endl;
    swapByReference(x, y);
    cout << "After swap by reference: x=" << x << " y=" << y << endl;

    return 0;
}
```

```cpp
int main() {
    int n;
    cout << "Enter number of elements: ";
    cin >> n;

    int arr[n];
    int* ptr = arr;

    cout << "Enter " << n << " Input numbers:\n";
    for (int i = 0; i < n; ++i)
        cin >> *(ptr + i);

    cout << "Output: ";
    for (int i = 0; i < n; ++i)
        cout << *(ptr + i) << " ";
    cout << endl;

    return 0;
}
```

```
7   void moveCursorLeft(int n = 1) {
8       cout << "\033[" << n << "D";
9   }
10
11  void moveCursorRight(int n = 1) {
12      cout << "\033[" << n << "C";
13  }
14
15  void clearLine() {
16      cout << "\033[2K\r";
17  }
18
19  // Line editor using dynamic char array with new[]
20  char* lineEditor(const char* prompt) {
21      int capacity = 16;              // initial buffer size
22      char* line = new char[capacity];
23      int length = 0;                 // number of characters entered
24      int cursorPos = 0;              // cursor position
25
26      cout << prompt;
27      cout.flush();
28
29      while (true) {
30          char ch = _getch();
31
32          if (ch == 13) {  // Enter
33              cout << endl;
34              break;
35          }
```

```
            cursorPos++;
        }
    }
    else if (ch == 8) {  // Backspace
        if (cursorPos > 0) {
            for (int i = cursorPos - 1; i < length - 1; i++)
                line[i] = line[i + 1];
            length--;
            cursorPos--;
            clearLine();
            cout << prompt;
            for (int i = 0; i < length; i++) cout << line[i];
            moveCursorLeft(length - cursorPos);
        }
    }
    else if (isprint(ch)) {  // Printable
        if (length + 1 >= capacity) {       // resize if needed
            int newCapacity = capacity * 2;
            char* newLine = new char[newCapacity];
            for (int i = 0; i < length; i++)
                newLine[i] = line[i];
            delete[] line;
            line = newLine;
            capacity = newCapacity;
        }
        for (int i = length; i > cursorPos; i--)
            line[i] = line[i - 1];
        line[cursorPos] = ch;
        cursorPos++;
```

```
                capacity = newCapacity;
            }
            for (int i = length; i > cursorPos; i--)
                line[i] = line[i - 1];
            line[cursorPos] = ch;
            cursorPos++;
            length++;
            clearLine();
            cout << prompt;
            for (int i = 0; i < length; i++) cout << line[i]
            moveCursorLeft(length - cursorPos);
        }
    }

    line[length] = '\0';
    return line;
}

int main() {
    system(""); // Enable ANSI on Windows
    char* name = lineEditor("Enter your name: ");
    if (name) {
        cout << "You entered: " << name << endl;
        delete[] name;
    }
    return 0;
}
```

```cpp
#include <iostream>
#include <memory>
using namespace std;

int main() {
    //Raw
    int val = 50;
    int* ptr = &val;
    int** ptr2 = &ptr;

    cout << "Value = " << val << endl;
    cout << "single pointer = " << *ptr << endl;
    cout << "pointer of pointer = " << **ptr2 << endl;
    //Modern
    auto val = make_shared<int>(50);
    shared_ptr<shared_ptr<int>> ptr2 = make_shared<shared_ptr<int>>(val);
    cout << "Value = " << **ptr2 << endl;
    return 0;
}
```

```cpp
#include <iostream>
#include <memory>
#include <string>
#include <conio.h>
using namespace std;

void gotoxy(short x, short y) {
    cout << "\033[" << y + 1 << ";" << x + 1 << "H";
}

void clearScreen() {
    cout << "\033[2J\033[H";
}

class Employee {
    string id;
    string name;
    string age;
    string salary;
    bool exists;

public:
    // --- Constructor ---
    Employee() : exists(false) {}

    // --- Setters ---
    void setId(const string& newId) { id = newId; }
    void setName(const string& newName) { name = newName; }
    void setAge(const string& newAge) { age = newAge; }
    void setSalary(const string& newSalary) { salary = newSalary; }
    void setExists(bool value) { exists = value; }

    // --- Getters ---
    string getId() const { return id; }
    string getName() const { return name; }
```

```
string getId() const { return id; }
string getName() const { return name; }
string getAge() const { return age; }
string getSalary() const { return salary; }
bool doesExist() const { return exists; }

void display() const {
    cout << "ID: " << id << "\n"
         << "Name: " << name << "\n"
         << "Age: " << age << "\n"
         << "Salary: " << salary << "\n";
}

void inputForm() {
    int choice = 0;
    char key;

    const int idLine = 2;
    const int nameLine = 3;
    const int ageLine = 4;
    const int salaryLine = 5;

    while (true) {
        clearScreen();
        cout << "=========== EMPLOYEE FORM ===========" << endl << endl;
        cout << "Id     : " << id << endl;
        cout << "Name   : " << name << endl;
        cout << "Age    : " << age << endl;
        cout << "Salary : " << salary << endl;
        cout << "\n(type to edit, Enter to save, ESC to cancel)\n";

        int cursorY;
        int cursorX = 9;
        switch (choice) {
            case 0: cursorY = idLine; cursorX += id.size(); break;
```

```
            gotoxy(cursorX, cursorY);
            key = _getch();

            if (key == -32) {
                key = _getch();
                if (key == 72 && choice > 0) choice--;
                else if (key == 80 && choice < 3) choice++;
            }
            else if (key == 27) { // ESC
                break;
            }
            else if (key == 13) { // ENTER
                exists = true;
                break;
            }
            else if (key == 8) { // BACKSPACE
                switch (choice) {
                    case 0: if (!id.empty()) id.pop_back(); break;
                    case 1: if (!name.empty()) name.pop_back(); break;
                    case 2: if (!age.empty()) age.pop_back(); break;
                    case 3: if (!salary.empty()) salary.pop_back(); break;
                }
            }
            else if (isprint(key)) { // Add printable characters
                switch (choice) {
                    case 0: id += key; break;
                    case 1: name += key; break;
                    case 2: age += key; break;
                    case 3: salary += key; break;
                }
            }
        }
    }
```

```
int main() {
    system(""); // enable ANSI escape codes

    int numEmployees;
    cout << "Enter number of employees to allocate: ";
    cin >> numEmployees;

    unique_ptr<Employee[]> employees(new Employee[numEmployees]);

    string menu[4] = {"New", "Display", "Display All", "Exit"};
    int choice = 0;
    char key;

    while (true) {
        clearScreen();
        cout << "--- Employee Menu ---\n\n";

        for (int i = 0; i < 4; i++) {
            if (choice == i)
                cout << "\033[31m> " << menu[i] << "\033[0m\n";
            else
                cout << "  " << menu[i] << "\n";
        }

        key = _getch();

        if (key == -32) {
            key = _getch();
            if (key == 72 && choice > 0) choice--;
            else if (key == 80 && choice < 3) choice++;
        }
        else if (key == 13) { // Enter
            clearScreen();

            if (choice == 0) { // New Employee
```

```
        emp.inputForm();

        if (emp.doesExist()) {
            int index;
            try {
                index = stoi(emp.getId());
            } catch (...) {
                cout << "Invalid ID. Must be a number.\n";
                _getch();
                continue;
            }

            if (index < 0 || index >= numEmployees)
                cout << "Invalid index. (0-" << numEmployees - 1 << ")\n";
            else {
                employees[index] = emp;
                employees[index].setExists(true);
                cout << "\nEmployee saved successfully at index " << index << ".\n";
            }
        } else cout << "\nCancelled.\n";
        _getch();
    }

    else if (choice == 1) { // Display One
        int index;
        cout << "Enter index (0-" << numEmployees - 1 << "): ";
        cin >> index;

        if (index < 0 || index >= numEmployees)
            cout << "Invalid index.\n";
        else if (employees[index].doesExist())
            employees[index].display();
        else
            cout << "Employee not found.\n";
```

```
                cout << "Employee not found.\n";

            cout << "\nPress any key...";
            _getch();
        }

        else if (choice == 2) { // Display All
            cout << "--- All Employees ---\n";
            bool any = false;
            for (int i = 0; i < numEmployees; ++i) {
                if (employees[i].doesExist()) {
                    any = true;
                    cout << "\nIndex " << i << ":\n";
                    employees[i].display();
                }
            }
            if (!any) cout << "No employees found.\n";
            cout << "\nPress any key...";
            _getch();
        }

        else if (choice == 3) { // Exit
            cout << "Exiting...\n";
            break;
        }
    }
}

    return 0;
}
```