Bom, não sabia de onde partiria para a implementação dessa intercalação balanceada, procurei tutoriais, para entender como funcionava. O que eu consegui dessas tentativas foi um código incompleto, que executa a ordenação, salva nas fitas, mas não salva até na metade das fitas e depois aloca um novo bloco lá na primeira, ele continua alocando nas fitas seguintes, e também estou com problemas na hora de exibir a saída final. Tenho o palpite que tenha haver com a quantidade de fitas que estou "criando", porque não específico quantas é para criar, ele só vai criando. Falta ter uma mudança no código de até quando ele vai intercalar as fitas, a gravação final no output.txt. Consegui pegar a ideia, mas falta desenvolver melhor o código. Abaixo terão prints da main do código, e das funções:

```cpp
#include "funcoes.cpp"

int main(){

    UINT CPAGE_UTF8 = 65001;
    UINT CPAGE_DEFAULT = GetConsoleOutputCP();
    SetConsoleOutputCP(CPAGE_UTF8);

    ifstream inputFile("input.txt"); // Substitua 'input.txt' pelo caminho para o arquivo de entrada
    ofstream outputFile("output.txt"); // Substitua 'output.txt' pelo arquivo de saída desejado

    if (!inputFile) {
        cerr << "Erro: não foi possível abrir o arquivo de entrada." << endl;
        return 1;
    }

    if (!outputFile) {
        cerr << "Erro: não foi possível criar o arquivo de saída." << endl;
        return 1;
    }

    createtapes(inputFile);

    inputFile.close();

    // Abra todos os arquivos de fita para mesclagem
    vector<ifstream> tapeFiles;

    for (int tapeNum = 0; ; ++tapeNum) {
        ifstream tapeFile("tape_" + to_string(tapeNum) + ".txt");
        if (!tapeFile)
            break;
        tapeFiles.push_back(move(tapeFile));
    }

    mergetapes(outputFile, tapeFiles);

    for (auto& file : tapeFiles) {
        if (file.is_open())
            file.close();
    }

    outputFile.close();

    cout << "Ordenação externa concluída. Os dados classificados são salvos em 'output.txt'." << endl;

    return 0;
}
```

```cpp
#include <iostream>
#include <fstream>
#include <cstring>
#include <vector>
#include <windows.h>
#include <algorithm>
#include <sstream>

const int TAPE_NUM = 6;
const int TAPE_SIZE = 3; // Defina o tamanho de cada fita (ajuste de acordo com o tamanho do conjunto de dados)
using namespace std;

// Função para mesclar dois arrays classificados

void merge(vector<string> &arr, int left, int mid, int right)
{
    // Calcula o tamanho dos subvetores a serem mesclados
    int n1 = mid - left + 1;
    int n2 = right - mid;

    // Cria vetores temporários para armazenar os subvetores
    vector<string> leftArr(n1);
    vector<string> rightArr(n2);

    // Copia os elementos dos subvetores originais para os temporários
    for (int i = 0; i < n1; ++i)
        leftArr[i] = arr[left + i];
    for (int i = 0; i < n2; ++i)
        rightArr[i] = arr[mid + 1 + i];

    // Inicializa índices para percorrer os subvetores
    int i = 0, j = 0, k = left;

    // Mescla os subvetores ordenadamente
    while (i < n1 && j < n2)
    {
        if (leftArr[i] <= rightArr[j])
            arr[k++] = leftArr[i++];
        else
            arr[k++] = rightArr[j++];
    }

    // Copia os elementos restantes do subvetor esquerdo (se houver)
    while (i < n1)
        arr[k++] = leftArr[i++];

    // Copia os elementos restantes do subvetor direito (se houver)
    while (j < n2)
        arr[k++] = rightArr[j++];
}

// Função para executar o merge sort
void mergeSort(vector<string> &arr, int left, int right)
{
    if (left < right)
    {
        // Calcula o ponto médio para dividir o vetor em subvetores
        int mid = left + (right - left) / 2;

        // Chama a função de mergeSort recursivamente para os subvetores
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

        // Mescla os subvetores ordenados
        merge(arr, left, mid, right);
    }
}
```

```cpp
// Função para ler dados do arquivo de entrada e criar fitas
void createtapes(ifstream &inputFile)
{
    vector<string> tape; // Vetor temporário para armazenar palavras de uma fita temporária.
    string line;         // Variável para armazenar uma linha lida do arquivo.

    int tapeNum = 0; // Número da fita atual.

    while (getline(inputFile, line))
    {                           // Lê cada linha do arquivo.
        istringstream iss(line); // Cria um stream de string a partir da linha.
        string word;            // Variável para armazenar cada palavra lida.
            while (iss >> word)
            {                       // Lê cada palavra da linha.
                tape.push_back(word); // Adiciona a palavra ao vetor da fita.
                if (tape.size() == TAPE_SIZE)
                {                                   // Verifica se a fita temporária atingiu o tamanho máximo.
                    mergeSort(tape, 0, TAPE_SIZE - 1); // Ordena a fita temporária.

                    ofstream tapeFile("tape_" + to_string(tapeNum) + ".txt", ofstream::app);
 // Abre arquivo para a fita ordenada.
                    for (const auto &w : tape)
                        tapeFile << w << " "; // Escreve cada palavra no arquivo.
                    tapeFile << "\t\t";
                    tapeFile.close();         // Fecha o arquivo.
                    tape.clear();             // Limpa o vetor da fita para a próxima.
                    ++tapeNum;                // Incrementa o número da fita.

                    if (tapeNum == 3)
                    { // Reinicia o número da fita quando atinge o limite.
                        tapeNum = 0;
                    }
                }
            }
    }

    // Se houver elementos restantes na última fita, ordena e salva em um arquivo.
    if (!tape.empty())
    {
        mergeSort(tape, 0, tape.size() - 1);
        ofstream tapeFile("tape_" + to_string(tapeNum) + ".txt");
        for (const auto &w : tape)
            tapeFile << w << " ";
        tapeFile.close();
    }
}

// Função para mesclar todas as fitas classificadas no arquivo de saída classificado final
void mergetapes(ofstream &outputFile, vector<ifstream> &tapeFiles)
{
    vector<string> currentValues(tapeFiles.size()); // Vetor para armazenar os valores atuais de cada fita.
    bool hasValue = true;                           // Indica se há valores a serem mesclados.

    while (hasValue)
    {                    // Enquanto houver valores a serem mesclados.
        hasValue = false; // Inicializa como falso, assume que não há mais valores.

        string smallestValue = "~"; // Valor inicialmente grande para comparar.

        int smallestIndex = -1; // Índice da fita com o menor valor.

        for (size_t i = 0; i < tapeFiles.size(); ++i)
        { // Para cada fita.
            if (tapeFiles[i] >> currentValues[i])
            {                    // Lê o próximo valor da fita.
                hasValue = true; // Há valores para mesclar.

                if (currentValues[i] < smallestValue)
                {                                          // Verifica se é o menor valor até agora.
                    smallestValue = currentValues[i];      // Atualiza o menor valor.
                    smallestIndex = static_cast<int>(i); // Atualiza o índice da fita com o menor valor.
                }
            }
        }

        if (hasValue)
        {                                     // Se há valores a serem mesclados.
            outputFile << smallestValue << " "; // Escreve o menor valor no arquivo de saída.

            if (!(tapeFiles[smallestIndex] >> currentValues[smallestIndex]))
            {                                    // Verifica se a fita está esgotada.
                tapeFiles[smallestIndex].close(); // Fecha o arquivo da fita.
            }
        }
    }
}
```