PROBLEMA INTERMEDIÁRIO 2

1 Análise Inicial

É pedido para desenvolver uma função que recebe uma lista de números e retorna a mesma lista sem números repetidos. Então, decidiu-se fazer uma lista auxiliar para guardar os valores não repetidos recebidos. Com isso, usaram-se dois laços para verificar quais valores já haviam sido adicionados na lista auxiliar e, ao final, retornou-se a lista auxiliar.

2 Funções

Primeiramente, criou-se uma lista auxiliar alocando espaço na memória igual ao tamanho da lista recebida, que deve ser enviado na chamada da função como um inteiro. Foi usada essa abordagem para a resolução do problema, pois ela foi considerada a melhor forma de evitar o acesso a posições inválidas do vetor. Imediatamente após isso, copiou-se o primeiro valor da lista recebida na auxiliar, já que ele sempre será um valor ”novo”, e guardou-se 1 na variável de tamanho da lista auxiliar.

```
double* filtra_diferentes(double *lista, int tam){ //Recebe uma lista de inteiros e
seu tamanho e retorna a lista sem valores repetidos
  int flag = 0, tam_aux = 0;
  double *aux = NULL, *ans = NULL;

  aux = (double*) malloc(tam*sizeof(double)); //Aloca o valor maximo da lista
auxiliar, que e igual ao tamanho da lista recebida

  aux[0] = lista[0]; //O primeiro valor sempre fara parte da lista auxiliar
  tam_aux = 1;
```

Figura 1: Criação e Inicialização da Lista Auxiliar

Então, usou-se um laço *for* para percorrer os valores da lista recebida, começando pelo segundo valor, já que o primeiro já foi adicionado na auxiliar. Dentro do laço, uma variável de flag é abaixada para cada novo valor observado e faz-se um novo laço para percorrer os valores dentro da lista auxiliar. Com isso, caso seja encontrado um valor igual na lista auxiliar, levanta-se a flag e quebra-se o segundo laço, pois o valor observado não é novo. Logo, se o laço for terminado e a flag ainda estiver abaixada, o valor observado é adicionado na lista auxiliar e a variável que guarda o tamanho da lista é incrementada em 1.

```
  for(int i = 1; i < tam; i++){ //Percorre a lista recebida a partir do segundo valor
    flag = 0;
    for(int j = 0; j < tam_aux; j++){ //Percorre a lista auxiliar
      if(aux[j] == lista[i]){ //Levanta o flag se encontrar algum valor que esta na
lista auxiliar
        flag = 1;
        break;
      }
    }
    if(!flag) aux[tam_aux++] = lista[i]; //Se o flag estiver abaixado, o valor ainda
nao havia aparecido e eh adicionado na lista auxiliar
  }
```

Figura 2: Armazenamento de Novos Valores na Lista Auxiliar

Finalmente, faz-se uma realocação do espaço da lista auxiliar para desocupar qualquer espaço desnecessário, usando a variável ”tam_aux”, que guarda o tamanho da lista auxiliar. Com isso, retorna-se o ponteiro da lista auxiliar, que é a lista filtrada apenas com valores distintos entre si.

```
  aux = realloc(aux, tam_aux*sizeof(double)); //Reduz o tamanho da lista auxiliar
para o tamanho necessario

  return aux; //Retorna a lista auxiliar
}
```

Figura 3: Realocação de Espaço e Retorno da Lista Filtrada