```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <time.h>
#include <pthread.h>

typedef struct block
{
    int link;
    int data;
    struct block *next;
} block;

void addBlock(int dataIn, int linkIn, block **mem);
block **defrag(block **arr);
block *blockInit(int dataIn, int linkIn);
void printArr(block **mem);

int main()
{
    block **mem;
    if ((mem = malloc(10000 * sizeof(block *))) == NULL)
    {
        printf("error creating array");
        return EXIT_FAILURE;
    }
    block *garbage = blockInit(0, -2);
    int x;
    for (x = 0; x < 10000; x++)
    {
        mem[x] = garbage;
    }
    int a = 0;
    int b = 0;
    printf("\nOriginal Memory\n");
    while (1)
    {
        if (scanf("%d %d", &a, &b) == 2)
        {
            addBlock(a, b, mem);
            printf("%d %d\n", a, b);
        }
        else
        {
            break;
        }
    }
```

```
    //printArr(mem);
    mem = defrag(mem);
    return EXIT_SUCCESS;
}
//Segfaulting when a block is being added to mem;
void addBlock(int dataIn, int linkIn, block **mem)
{
    block *cur = blockInit(dataIn, linkIn);
    mem[cur->data] = cur;
    if (cur->link != -1 || cur->link != -2)
    {
        cur->next = mem[cur->link];
    }
}
block **defrag(block **arr)
{
    int x = 0;
    int y;
    int end = 9999;
    int fake;
    int bMove = 0;
    int bemptyMove = 0;
    block **newArr;
    block *linked;
    int check[10000];
    if ((newArr = malloc(10000 * sizeof(block *))) == NULL)
    {
        printf("error creating array");
        return arr;
    }
    for (fake = 0; fake < 10000; fake++)
    {
        block *ph = blockInit(fake, -2);
        check[fake] = 0;
        newArr[fake] = ph;
    }
    while (x < 10000 && y < 10000)
    {
        if (arr[y]->link != -2 && check[y] < 1)
        {
            newArr[x] = arr[y];
            if (arr[y]->data != x)
            {
                bMove++;
            }
            newArr[x]->data = x;
            if (arr[y]->link != -1)
            {
```

```
            newArr[x]->link = x + 1;
        }
        check[y] = 1;
        x++;
        if (arr[y]->link != -1)
        {
            linked = arr[arr[y]->link];
            //Follow the chain;
            while (linked != NULL)
            {
                if (check[linked->link] < 0)
                {
                    if (linked->link != -1 && linked->link != -2)
                    {
                        if (linked->data != x)
                        {
                            bMove++;
                        }
                        check[linked->data] = 1;
                        linked->data = x;
                        newArr[x] = linked;
                        linked = arr[linked->link];
                        newArr[x]->link = x + 1;
                        x++;
                    }
                    else if (linked->link != -2)
                    {
                        if (linked->data != x)
                        {
                            bMove++;
                        }
                        check[linked->data] = 1;
                        linked->data = x;
                        newArr[x] = linked;
                        newArr[x]->link = -1;
                        x++;
                        linked = NULL;
                        break;
                    }
                    else
                    {
                        block *filler = blockInit(end, -2);
                        check[linked->data] = 1;
                        if (arr[y]->data != end)
                        {
                            bemptyMove++;
                        }
                        newArr[end] = filler;
```

```
                            end--;
                            linked == NULL;
                            break;
                        }
                    }
                    else
                    {
                        break;
                    }
                }
            }
        }
        else if (arr[y]->link == -2)
        {
            check[y] = 1;
            block *filler = blockInit(end, -2);
            if (arr[y]->data != end)
            {
                bemptyMove++;
            }
            newArr[end] = filler;
            end--;
        }
        y++;
    }
    int h;
    for (h = 0; h < 10000; h++)
    {
        if (newArr[h]->link == -2)
        {
            newArr[h]->link = -1;
            break;
        }
    }
    printArr(newArr);
    printf("\nTotal blocks moved (Excluding the movement of empty blocks): %d\nTotal
blocks moved (Including the movement of empty blocks): %d\n", bMove, bMove +
bemptyMove);
    return newArr;
}
block *blockInit(int dataIn, int linkIn)
{
    block *cur;
    if ((cur = malloc(sizeof(block))) == NULL)
    {
        printf("\nwas unable to create a new memory block\n");
    }
    cur->data = dataIn;
```

```
    cur->link = linkIn;
    cur->next = NULL;
    return cur;
}
void printArr(block **arr)
{
    int p;
    for (p = 0; p < 10000; p++)
    {
        printf("\n%d\t%d", arr[p]->data, arr[p]->link);
    }
    printf("\n");
}
```