Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный исследовательский университет "Высшая школа экономики"

Московский институт электроники и математики им. А.Н. Тихонова НИУ ВШЭ
Департамент компьютерной инженерии

Курс: Алгоритмизация и программирование

Раздел	**Макс. оценка**	**Итоговая оценка**
Работа программы	1	
Тесты	1	
Правильность алгоритма	3	
Ответы на вопросы	2	
Дополнительное задание	3	
Итого	10	

Отчет по лабораторной работе № 7

Студент: Чапайкин Арсений Георгиевич

Группа: БИВ242

Вариант: № 171 (3, 7)

Руководитель: Елисеенко А.М.

Оценка:

Дата сдачи:

МОСКВА 2024

Содержание

Задание **2**

Листинг программы **3**

Распечатка тестов к программе и результатов **6**

Задание

1. Создать связанный список для хранения целых чисел. Число записей неизвестно, данные читаются из файла, имя которого задает пользователь. Имя выходного файла также задается пользователем. Входной файл содержит данные для исходного списка, выходной – для конечного списка. Построить на этом списке стек.

2. Удалить из списка все четные элементы.

3. Удалить из списка все элементы, расположенные между первым максимальным и последним минимальным элементом (необходимо освободить память, занимаемую удаляемыми элементами).

Листинг программы

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

// Stack implementation
typedef struct Node {
    int value;
    struct Node* next;
} Node;

Node* push(Node* head, int value) {
    Node* temp = (Node*)malloc(sizeof(Node));
    temp->value = value;
    temp->next = head;
    head = temp;
    return head;
}

Node* pop(Node* head) {
    if (head == NULL) {
        return NULL;
    }
    Node* temp = head;
    head = head->next;
    free(temp);
    return head;
}

void printContents(FILE* ostream, Node* head) {
    for (Node* i = head; i != NULL; i = i->next) {
        fprintf(ostream, "%d ", i->value);
    }
    fprintf(ostream, "\n");
}

// Deletes interval [left, right)
void deleteInterval(Node* left, Node* right) {
    for (Node* i = left; i != right;) {
        Node* temp = i;
        i = i->next;
        free(temp);
    }
}

// Utility functions
void inputString(char** string, char stopchar) {
    size_t size = 1, index = 0;
    *string = (char*)malloc(sizeof(char));

    char c = EOF;
    while ((c = getc(stdin)) != EOF && c != stopchar) {
        (*string)[index++] = c;
        if (index == size - 1) {
            size *= 2;
            *string = realloc(*string, size);
        }
    }
```

```
58
59      (*string)[index] = '\0';
60      *string = realloc(*string, index + 1);
61  }
62
63  void inputStack(FILE* istream, Node** stack) {
64      int value;
65      while (fscanf(istream, "%d[^;\n]", &value) == 1) {
66          fscanf(istream, "%*c");
67          *stack = push(*stack, value);
68      }
69  }
70
71  // Task 2
72  Node* deleteAllEven(Node* head) {
73      Node* cur = head;
74      Node* prev = NULL;
75      while (cur != NULL) {
76          if (cur->value % 2 == 0) {
77              Node* temp = cur;
78              cur = cur->next;
79              free(temp);
80          } else {
81              if (prev == NULL) {
82                  head = cur;
83              } else {
84                  prev->next = cur;
85              }
86              prev = cur;
87              cur = cur->next;
88          }
89      }
90      if (prev != NULL) {
91          prev->next = NULL;
92          return head;
93      } else {
94          return NULL;
95      }
96  }
97
98  // Task 3
99  Node* deleteAllBetweenFirstMaxAndLastMin(Node* head) {
100     Node* max = head;
101     Node* min = head;
102     bool minIsBigger = true;
103     for (Node* i = head; i != NULL; i = i->next) {
104         if (i->value > max->value) {
105             max = i;
106             minIsBigger = false;
107         }
108         if (i->value <= min->value) {
109             min = i;
110             minIsBigger = true;
111         }
112     }
113     if (max != min) {
114         if (minIsBigger) {
115             deleteInterval(max->next, min);
116             max->next = min;
117         } else {
```

```
            deleteInterval(min->next, max);
            min->next = max;
        }
    }
    return head;
}

int main(void) {
    char* inputFileName;
    char outputFileName[] = "output.txt";
    inputString(&inputFileName, '\n');

    FILE* istream = fopen(inputFileName, "r");
    FILE* ostream = fopen(outputFileName, "w");

    if (istream == NULL) {
        fprintf(ostream, "No file called '%s'", inputFileName);
        return 1;
    }

    Node* stack = NULL;
    inputStack(istream, &stack);

    fputs("Input:\n", ostream);
    printContents(ostream, stack);

    stack = deleteAllEven(stack);
    if (stack == NULL) {
        fputs("Stack became empty after all even elements were deleted\n", ostream)
            ;
        return 0;
    }

    fputs("Result of task 2:\n", ostream);
    printContents(ostream, stack);

    stack = deleteAllBetweenFirstMaxAndLastMin(stack);
    fputs("Result of task 3:\n", ostream);
    printContents(ostream, stack);

    deleteInterval(stack, NULL);
    fclose(istream);
    fclose(ostream);

    return 0;
}
```

Распечатка тестов к программе и результатов

Номер	Исходные данные	Результат
1	1;2;3;4;5;6	Input: 6 5 4 3 2 1 Result of task 2: 5 3 1 Result of task 3: 5 1
2	0;2;0;4;0;6	Input: 6 0 4 0 2 0 Stack became empty after all even elements were deleted
3	5;2;0;4;0;1	Input: 1 0 4 0 2 5 Result of task 2: 1 5 Result of task 3: 1 5
4	1;2;0;4;0;5	Input: 5 0 4 0 2 1 Result of task 2: 5 1 Result of task 3: 5 1
5	5;2;0;4;0;0	Input: 0 0 4 0 2 5 Result of task 2: 5 Result of task 3: 5
5	0;2;0;4;0;5	Input: 5 0 4 0 2 0 Result of task 2: 5 Result of task 3: 5