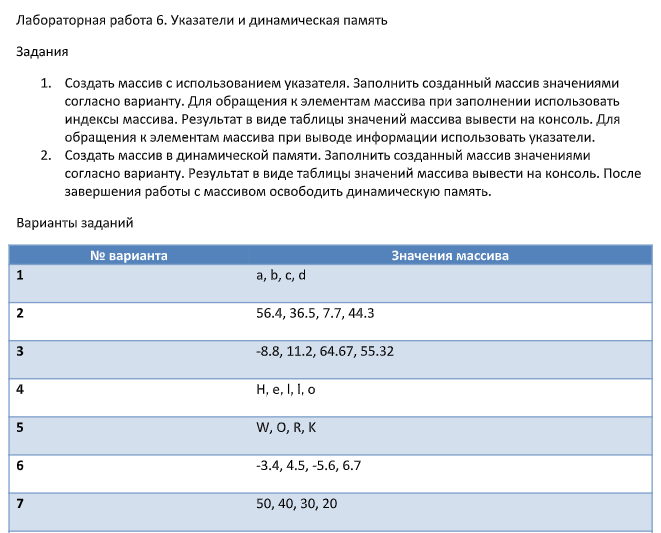
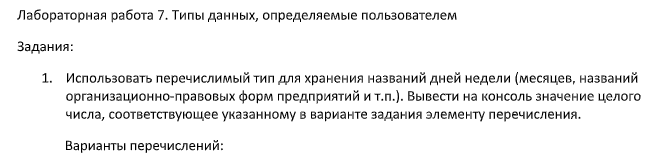
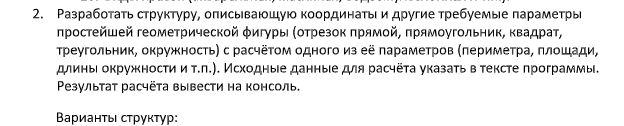
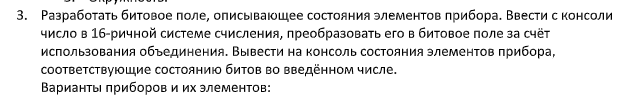
#include <stdio.h>  
#include <math.h>  
#include <stdlib.h>  
  
int main() {  
 int x[7] = {90, 76, 54, 23, 56, 12, 48};  
 printf("\nidx : val\n");   
 for (int i = 0; i < sizeof(x) / sizeof(int); i++) {  
 printf("%d : %d\n", i, x[i]); //Выводим массив  
 }  
  
 int a[2][2] = {{2, 5}, {2, 2}}; //Задаем матрицы  
 int b[2][2] = {{1, 2}, {0, 1}};  
 int c[2][2] = {{0, 0}, {0, 0}};  
 for (int i = 0; i < 2; i++) {  
 for (int j = 0; j < 2; j++) {  
 for (int h = 0; h < 2; h++) {  
 c[i][j] += a[i][h] \* b[h][j]; //Перемножаем  
 }  
 }  
 }  
 printf("\n");  
 for (int i = 0; i < 2; i++) {  
 for (int j = 0; j < 2; j++) {  
 printf("%d ", c[i][j]); //Выводим результат перемножения матриц  
 }  
 printf("\n");  
 }  
 return 0;  
}

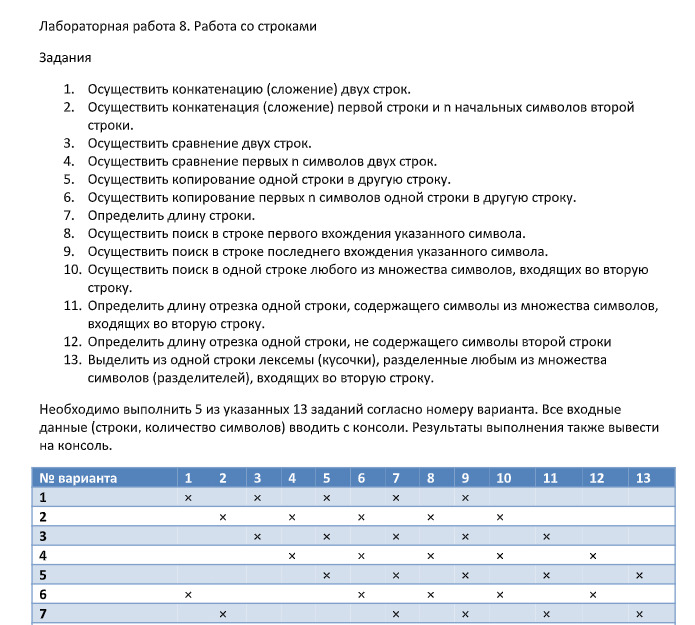
#include <stdio.h>  
#include <math.h>  
#include <stdlib.h>  
#include <string.h>  
  
int main() {  
 int ar[4]; //Создаем массив  
 int \*p = (int \*) &ar; //Создаем указатель на массив  
 ar[0] = 50; ar[1] = 40; ar[2] = 30; ar[3] = 20; //Заполняем массив  
 printf("idx : val\n");  
 for (int i = 0; i < 4; i++) {  
 printf("%d : %d\n", i, p[i]); //Выводим его по указателям  
 }  
 printf("\n");  
  
 int \*a;  
 a = calloc(4, sizeof(int)); //Создаем динамически массив  
 a[0] = 50; a[1] = 40; a[2] = 30; a[3] = 20;  
 printf("\nidx : val\n");  
 for (int i = 0; i < 4; i++) {  
 printf("%d : %d\n", i, a[i]); //Выводим его  
 }  
 free(a);  
 return 0;  
}







#include <stdio.h>  
  
int main() {  
 enum sweet {*pie*, *cake*, *brownie*, *icecream*, *candy*}; //Инициализирую данные перечисляемого типа  
 enum sweet sweets = *candy*; //Объявляю sweets типа sweet  
 printf("%d\n", sweets); // Вывожу число, соответсвующее конфете  
  
 struct S {  
 double x, y, a; //Объявляем структуру квадрата: расположение левого нижнего угла  
 }; //и длина его стороны  
 struct S square = {0, 0, 5.2}; //Объявляем переменную класса S  
 double perimeter = square.a \* 4; //Вычисляем его периметр  
 printf("%lf\n", perimeter);  
  
 union Printer {  
 struct laser\_printer {  
 unsigned elements : 1; //Отводим по одному биту на состояния в структуре  
 unsigned low\_toner : 1;  
 unsigned broken : 1;  
 unsigned no\_paper : 1;  
 } stats;  
 unsigned input; //Объединяем состояния с input  
 } printer;  
 scanf("%x", &printer.input);  
 printf("Элементы: %s\n", printer.stats.elements ? "ON" : "OFF");  
 printf("Мало тонера: %s\n", printer.stats.low\_toner ? "ON" : "OFF");  
 printf("Сломан: %s\n", printer.stats.broken ? "ON" : "OFF");  
 printf("Нет бумаги: %s\n", printer.stats.no\_paper ? "ON" : "OFF");  
 return 0;  
}

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
  
#define PI acos(-1.)  
  
int main() {  
 ///lab rab 8  
 int n = 50;  
 char \*str1 = (char \*) malloc(sizeof(char) \* n);  
 char \*str2 = (char \*) malloc(sizeof(char) \* n);  
 char \*str3 = (char \*) malloc(sizeof(char) \* n);  
 char \*str5 = (char \*) malloc(sizeof(char) \* n);  
 char \*str7 = (char \*) malloc(sizeof(char) \* n);  
 char \*str8 = (char \*) malloc(sizeof(char) \* n);  
 char \*str9 = (char \*) malloc(sizeof(char) \* n);  
 char \*str0 = (char \*) malloc(sizeof(char) \* n);  
  
 //2. Осуществить конкатенация (сложение) первой строки и n начальных символов второй строки.  
 scanf("%s", str1);  
 scanf("%s", str2);  
 int x;  
 scanf("%d", &x);  
 char \*temp = (char \*) malloc(sizeof(char) \* x);  
 strncpy(temp, str2, x); //temp – первые x символов со второй строки  
 strcat(str1, temp); //Объединяем первую строку и первые x символов второй строки  
 printf("%s\n", str1);  
  
 //7. Определить длину строки.  
 scanf("%s", str3);  
 printf("%d\n", strlen(str3)); //Вычисляем длину  
  
 //9. Осуществить поиск в строке последнего вхождения указанного символа.  
 char c;  
 scanf("%s", str5);  
 c = getc(stdin); //Считываем пробел  
 c = getc(stdin); //Считываем символ  
 printf("%d\n", strrchr(str5, c) - str5 + 1); //Находим правое вхождение  
  
 //11. Определить длину отрезка одной строки, содержащего символы из множества символов, входящих во вторую строку.  
 scanf("%s", str7);  
 scanf("%s", str8);  
 int ans = 0, cur = 0;  
 for (int i = 0; i < strlen(str7); i++) {  
 int flag = 0;  
 for (int j = 0; j < strlen(str8); j++) {  
 if (str7[i] == str8[j]) {  
 flag = 1;  
 break;  
 }  
 }  
 if (flag) cur++; // Если текущий символ в строке str7 равен хоть какому из str8,  
 else cur = 0; //то увеличиваем длину нашего отрезка, иначе обрезаем  
  
 if (ans < cur) ans = cur;  
 }  
 printf("%d\n", ans);  
  
 //13. Выделить из одной строки лексемы (кусочки), разделенные любым из множества символов (разделителей), входящих во вторую строку.  
 scanf("%s", str9);  
 scanf("%s", str0);  
 int l = 0;  
 for (int i = 0; i < strlen(str9); i++) {  
 int flag = 0;  
 for (int j = 0; j < strlen(str0); j++) {  
 if (str9[i] == str0[j]) {  
 flag = 1;  
 break;  
 }  
 }  
 if (flag) { //Выводим строку, если текущий символ - разделитель  
 temp = (char \*) malloc(sizeof(char) \* (i – l));  
 strncpy(temp, str9 + l, i - l);  
 printf("%s\n", temp);  
 l = i + 1;  
 }  
 }  
 return 0;  
}