Qno.1)

#include <stdio.h>

#define MAX\_DISHES 100 // maximum number of Petri dishes

int main() {

int n; // number of Petri dishes

int label[MAX\_DISHES], orig[MAX\_DISHES], newn[MAX\_DISHES]; // Petri dish data

double pr[MAX\_DISHES]; // reproduction rate

int a\_labels[MAX\_DISHES], b\_labels[MAX\_DISHES]; // labels of A and B Petri dishes

int na = 0, nb = 0; // number of A and B Petri dishes

// input Petri dish data

printf("Enter total number of Petri dishes: ");

scanf("%d", &n);

printf("Enter Petri dish label, original bacterial number, new bacterial number after one hour reproduction:\n");

for (int i = 0; i < n; i++) {

scanf("%d %d %d", &label[i], &orig[i], &newn[i]);

pr[i] = (double)newn[i] / orig[i];

}

// classify Petri dishes into A and B sub-species

for (int i = 0; i < n; i++) {

int is\_a = 1; // assume it's A

for (int j = 0; j < n; j++) {

if (i != j && pr[i] - pr[j] > pr[j] / 1000.0) {

// if PR difference is larger than 0.1%, it's not A

is\_a = 0;

break;

}

}

if (is\_a) {

a\_labels[na++] = label[i];

} else {

b\_labels[nb++] = label[i];

}

}

// output results

printf("Running results:\n");

printf("%d in A sub-species and Petri dish labels from smaller PR to bigger PR are", na);

for (int i = 0; i < na; i++) {

int minj = i;

for (int j = i + 1; j < na; j++) {

if (pr[a\_labels[j]-1] < pr[a\_labels[minj]-1]) {

minj = j;

}

}

int temp = a\_labels[i];

a\_labels[i] = a\_labels[minj];

a\_labels[minj] = temp;

printf(" %d", a\_labels[i]);

}

printf("\n");

printf("%d in B sub-species and Petri dish labels from smaller PR to bigger PR are", nb);

for (int i = 0; i < nb; i++) {

int minj = i;

for (int j = i + 1; j < nb; j++) {

if (pr[b\_labels[j]-1] < pr[b\_labels[minj]-1]) {

minj = j;

}

}

int temp = b\_labels[i];

b\_labels[i] = b\_labels[minj];

b\_labels[minj] = temp;

printf(" %d", b\_labels[i]);

}

printf("\n");

return 0;

}

Qno.2)

#include <stdio.h>

#include <stdlib.h>

int main() {

int n; // number of integers

int max\_odd = -1, min\_even = -1; // maximum odd and minimum even seen so far

int x; // current integer

// input integers and find maximum odd and minimum even

printf("Enter a series of integer numbers: ");

while (scanf("%d", &x) == 1) {

if (x % 2 == 1 && x > max\_odd) {

max\_odd = x;

}

if (x % 2 == 0 && (min\_even == -1 || x < min\_even)) {

min\_even = x;

}

getchar(); // consume the newline character

}

// compute and output absolute value of difference

if (max\_odd == -1 || min\_even == -1) {

printf("No odd and even numbers found\n");

} else {

int diff = abs(max\_odd - min\_even);

printf("Result is: %d\n", diff);

}

return 0;

}

Qno.3) #include <stdio.h>

int main() {

int n, m; // number of monkeys and value of m

int pos = 0; // current position in circle

// input number of monkeys and value of m

printf("Enter total number of monkeys in a group: ");

scanf("%d", &n);

printf("Enter m value: ");

scanf("%d", &m);

// simulate elimination process

for (int i = 1; i < n; i++) {

pos = (pos + m) % (n - i + 1);

}

// output result

printf("The king will be %d\n", pos);

return 0;

}

Qno.4)

#include <stdio.h>

int main() {

int payment, remaining; // payment and remaining amount

int bills[6] = {100, 50, 20, 10, 5, 1}; // available bills

int counts[6] = {0, 0, 0, 0, 0, 0}; // counts of each bill

// input payment

printf("Enter total payment: ");

scanf("%d", &payment);

remaining = payment;

// compute minimum number of bills

for (int i = 0; i < 6; i++) {

counts[i] = remaining / bills[i];

remaining = remaining % bills[i];

}

// output results

printf("Result of minimum number of bills: %d\n", payment);

for (int i = 0; i < 6; i++) {

printf("$%d bill: %d\n", bills[i], counts[i]);

}

return 0;

}