

Celem zadania jest zaimplementowanie serwera i klienta gry w kierki. Serwer przeprowadza grę. Klienty reprezentują graczy.

Zasady gry

W kierki gra czterech graczy standardową 52-kartową talią. Gracze siedzą przy stole na miejscach N (ang. *north*), E (ang. *east*), S (ang *south*), W (ang. *west*). Rozgrywka składa się z rozdań. W każdym rozdaniu każdy z graczy otrzymuje na początku po 13 kart. Gracz zna tylko swoje karty. Gra składa się z 13 lew. W pierwszej lewie wybrany gracz wychodzi, czyli rozpoczyna rozdanie, kładąc wybraną swoją kartę na stół. Po czym pozostali gracze w kolejności ruchu wskazówek zegara dokładają po jednej swojej karcie. Istnieje obowiązek dokładania kart do koloru. Jeśli gracz nie ma karty w wymaganym kolorze, może położyć kartę w dowolnym innym kolorze. Nie ma obowiązku przebijania kartą starszą. Gracz, który wyłożył najstarszą kartę w kolorze karty położonej przez gracza wychodzącego, bierze lewę i wychodzi jako pierwszy w następnej lewie. Obowiązuje standardowe starszeństwo kart (od najsłabszej): 2, 3, 4, ..., 9, 10, walet, dama, król, as.

W grze chodzi o to, żeby brać jak najmniej kart. Za branie kart otrzymuje się punkty. Wygrywa gracz, który w całej rozgrywce zbierze najmniej punktów. Jest siedem typów rozdań:

- nie brać lew, za każdą wziętą lewę dostaje się 1 punkt;
- nie brać kierów, za każdego wziętego kiera dostaje się 1 punkt;
- nie brać dam, za każdą wziętą damę dostaje się 5 punktów;
- nie brać panów (waletów i króli), za każdego wziętego pana dostaje się 2 punkty;
- nie brać króla kier, za jego wzięcie dostaje się 18 punktów;
- nie brać siódmej i ostatniej lewy, za wzięcie każdej z tych lew dostaje się po 10 punktów;
- rozbójnik, punkty dostaje się za wszystko wymienione powyżej.

Rozdania nie trzeba rozgrywać do końca – można je przerwać, jeśli już wiadomo, że wszystkie punkty zostały rozdysponowane.

Parametry wywołania serwera

Parametry wywołania serwera mogą być podawane w dowolnej kolejności. Jeśli parametr został podany więcej niż raz, to obowiązuje jego pierwsze lub ostatnie wystąpienie na liście parametrów.

−p <port>

Określa numer portu, na którym ma nasłuchiwać serwer. Parametr jest opcjonalny. Jeśli nie został podany lub ma wartość zero, to wybór numeru portu należy scedować na wywołanie funkcji **bind**.

−f <file>

Określa nazwę pliku zawierającego definicję rozgrywki. Parametr jest obowiązkowy.

−t <timeout>

Określa maksymalny czas w sekundach oczekiwania serwera. Jest to liczba dodatnia. Parametr jest opcjonalny. Jeśli nie podano tego parametru, czas ten wynosi 5 sekund.

Parametry wywołania klienta

Parametry wywołania klienta mogą być podawane w dowolnej kolejności. Jeśli parametr został podany więcej niż raz lub podano sprzeczne parametry, to obowiązuje pierwsze lub ostatnie wystąpienie takiego parametru na liście parametrów.

−h <host>

Określa adres IP lub nazwę hosta serwera. Parametr jest obowiązkowy.

−p <port>

Określa numer portu, na którym nasłuchuje serwer. Parametr jest obowiązkowy.

−4

Wymusza w komunikacji z serwerem użycie IP w wersji 4. Parametr jest opcjonalny.

−6

Wymusza w komunikacji z serwerem użycie IP w wersji 6. Parametr jest opcjonalny.

Jeśli nie podano ani parametru **−4**, ani **−6**, to wybór wersji protokołu IP należy scedować na wywołanie funkcji **getaddrinfo**, podając **ai_family = AF_UNSPEC**.

−N

−E

−S

−W

Określa miejsce, które klient chce zająć przy stole. Parametr jest obowiązkowy.

–a
kresla miejsce, które klient chce zająć przy stole. Parametr jest obowiązkowy.

–a





Parametr jest opcjonalny. Jeśli jest podany, to klient jest automatycznym graczem. Jeśli nie jest podany, to klient jest pośrednikiem między serwerem a graczem-użytkownikiem.

Protokół komunikacyjny

Serwer i klient komunikują się za pomocą TCP. Komunikaty są napisami ASCII zakończonymi sekwencją `\r\n`. Oprócz tej sekwencji w komunikatach nie ma innych białych znaków. Komunikaty nie zawierają terminalnego zera. Miejsce przy stole koduje się literą **N**, **E**, **S** lub **W**. Typ rozdania koduje się cyfrą od **1** do **7**. Numer lewy koduje się liczbą od **1** do **13** zapisaną przy podstawie 10 bez zer wiodących. Przy kodowaniu kart najpierw podaje się wartość karty:

- **2**, **3**, **4**, ..., **9**, **10**, **J**, **Q**, **K**, **A**.

Następnie podaje się kolor karty:

- **C** – , trefl, żołądz (ang. *club*),
- **D** – , karo, dzwonek (ang. *diamond*),
- **H** – , kier, czerwień (ang. *heart*),
- **S** – , pik, wino (ang. *spade*).

Serwer i klient przesyłają następujące komunikaty.

IAM<miejsce przy stole>\r\n

Komunikat wysyłany przez klienta do serwera po nawiązaniu połączenia. Informuje, które miejsce przy stole chce zająć klient. Jeśli klient nie przyśle takiego komunikatu w czasie **timeout**, serwer zamyka połączenie z tym klientem. W ten sposób serwer traktuje również klienta, który po nawiązaniu połączenia przysłał błędny komunikat.

BUSY<lista zajętych miejsc przy stole>\r\n

Komunikat wysyłany przez serwer do klienta, jeśli wybrane miejsce przy stole jest już zajęte. Jednocześnie informuje go, które miejsca przy stole są zajęte. Po wysłaniu tego komunikatu serwer zamyka połączenie z klientem. W ten sposób serwer traktuje również klienta, który próbuje podłączyć się do trwającej rozgrywki.

DEAL<typ rozdania><miejsce przy stole klienta wychodzącego jako pierwszy w rozdaniu><lista kart>\r\n

Komunikat wysyłany przez serwer do klientów po zebraniu się czterech klientów. Informuje o rozpoczęciu rozdania. Lista zawiera 13 kart, które klient dostaje w tym rozdaniu.

TRICK<numer lewy><lista kart>\r\n

Komunikat wysyłany przez serwer do klienta z prośbą o położenie karty na stole. Lista kart zawiera od zera do trzech kart aktualnie leżących na stole. Jeśli klient nie odpowie w czasie **timeout**, to serwer ponawia prośbę. Komunikat wysyłany przez klienta do serwera z kartą, którą klient kładzie na stole (lista kart zawiera wtedy jedną kartę).

WRONG<numer lewy>\r\n

Komunikat wysyłany przez serwer do klienta, który przysłał błędny komunikat w odpowiedzi na komunikat **TRICK**. Wysyłany również wtedy, gdy klient próbuje położyć kartę na stole nieproszony o to. Zawartość błędnego komunikatu jest ignorowana przez serwer.

TAKEN<numer lewy><lista kart><miejsce przy stole klienta biorącego lewę>\r\n

Komunikat wysyłany przez serwer do klientów. Informuje, który z klientów wziął lewę. Lista kart zawiera cztery karty składające się na lewę w kolejności, w jakiej zostały położone na stole.

SCORE<miejsce przy stole klienta><liczba punktów><miejsce przy stole klienta><liczba punktów><miejsce przy stole klienta><liczba punktów><miejsce przy stole klienta><liczba punktów>\r\n

Komunikat wysyłany przez serwer do klientów po zakończeniu rozdania. Informuje o punktacji w tym rozdaniu.

TOTAL<miejsce przy stole klienta><liczba punktów><miejsce przy stole klienta><liczba punktów><miejsce przy stole klienta><liczba punktów><miejsce przy stole klienta><liczba punktów>\r\n

Komunikat wysyłany przez serwer do klientów po zakończeniu rozdania. Informuje o łącznej punktacji w rozgrywce.

Klient ignoruje błędne komunikaty od serwera.

Po zakończeniu rozgrywki serwer rozłącza wszystkie klienty i kończy działanie. Po rozłączeniu się serwera klient kończy działanie.

Jeśli klient rozłączy się w trakcie rozgrywki, to serwer zawiesza rozgrywkę w oczekiwaniu na podłączenie się klienta na puste miejsce przy stole. Po podłączeniu się klienta serwer przekazuje mu stan aktualnego rozdania. Za pomocą komunikatu **DEAL** przekazuje karty, które klient dostał w tym rozdaniu. Za pomocą komunikatów **TAKEN** przekazuje dotychczas rozegrane lewy. Następnie serwer wznawia rozgrywkę i wymianę komunikatów **TRICK**.

Wymagania funkcjonalne

Programy powinny dokładnie sprawdzać poprawność parametrów wywołania. Programy powinny wypisywać zrozumiałe komunikaty o błędach na standardowe wyjście diagnostyczne.

Zakładamy, że serwer jest uczciwy. Natomiast klienty nie muszą być uczciwe. Serwer powinien dokładnie sprawdzać, czy klienty spełniają zasady gry.

W kliencie należy zaimplementować jakąś heurystyczną strategię gry.

Serwer oraz klient działający jako gracz automatyczny wypisują na standardowe wyjście raport z rozgrywki.

Programy kończą się kodem 0, jeśli rozgrywka przebiegła do końca, a w przeciwnym przypadku kodem 1.

Format pliku definiującego rozgrywkę

Plik o nazwie podanej serwerowi zawiera tekstowy opis rozgrywki. Opisane są w nim po kolei rozdania do rozegrania. Opis każdego rozdania składa się z pięciu linii opisujących typ rozdania, klienta wychodzącego jako pierwszy w rozdaniu i karty rozdawane poszczególnym klientom:

```
<typ rozdania><miejsce przy stole klienta wychodzącego jako pierwszy w rozdaniu>\n
<lista kart klienta N>\n
<lista kart klienta E>\n
<lista kart klienta S>\n
<lista kart klienta W>\n
```

Wolno założyć, że zawartość tego pliku jest poprawna.

Format raportu z rozgrywki

Raport z rozgrywki zawiera po kolei **wszystkie** wysłane i odebrane komunikaty (również te błędne). Komunikat poprzedza się, podanymi w nawiasach kwadratowych, adresem IP i numerem portu nadawcy, adresem IP i numerem portu odbiorcy oraz czasem odebrania lub wysłania.

Na przykład, jeśli serwer 11.22.33.44 na porcie 1234 dostał komunikat **IAMN\r\n** od klienta 44.44.44.44 z portu 4321 i odesłał komunikat **BUSYNW\r\n**, to powinien wypisać:

```
[44.44.44.44:4321,11.22.33.44:1234,2024-04-25T18:21:00.000] IAMN\r\n
[11.22.33.44:1234,44.44.44.44:4321,2024-04-25T18:21:00.010] BUSYNW\r\n
```

Komunikacja klienta z użytkownikiem

Klient działający jako pośrednik udostępnia użytkownikowi interfejs tekstowy. Interfejs użytkownika powinien być intuicyjny. Klient wypisuje na standardowe wyjście informacje dla użytkownika i prośby od serwera o położenie karty. Klient czyta ze standardowego wejścia decyzje i polecenia użytkownika, na przykład polecenie wyświetlenia kart na ręce i wziętych lew. Klient w każdej chwili powinien móc spełnić polecenie użytkownika. Komunikacja z serwerem nie może blokować interfejsu użytkownika.

Informacje od serwera formatowane są następująco:

```
BUSY<lista zajętych miejsc przy stole>
Place busy, list of busy places received: <lista zajętych miejsc przy stole>.
```

```
DEAL<typ rozdania><miejsce przy stole klienta wychodzącego jako pierwszy w rozdaniu><lista kart>
New deal <typ rozdania>: staring place <miejsce przy stole klienta wychodzącego jako pierwszy w rozdaniu>, your cards: <lista kart>.
```

```
WRONG<numer lewy>
Wrong message received in trick <numer lewy>.
```

```
TAKEN<numer lewy><lista kart><miejsce przy stole klienta biorącego lewę>
A trick <numer lewy> is taken by <miejsce przy stole klienta biorącego lewę>, cards <lista kart>.
```

```
SCORE<miejsce przy stole klienta><liczba punktów><miejsce przy stole klienta><liczba punktów><miejsce przy stole klienta><liczba punktów><miejsce przy stole klienta><liczba punktów>
The scores are:
<miejsce przy stole klienta> | <liczba punktów>
<miejsce przy stole klienta> | <liczba punktów>
<miejsce przy stole klienta> | <liczba punktów>
<miejsce przy stole klienta> | <liczba punktów>
```

```
TOTAL<miejsce przy stole klienta><liczba punktów><miejsce przy stole klienta><liczba punktów><miejsce przy stole klienta><liczba punktów><miejsce przy stole klienta><liczba punktów>
The total scores are:
<miejsce przy stole klienta> | <liczba punktów>
<miejsce przy stole klienta> | <liczba punktów>
<miejsce przy stole klienta> | <liczba punktów>
<miejsce przy stole klienta> | <liczba punktów>
```

```
TRICK<numer lewy><lista kart>
Trick: (<numer lewy>) <lista kart>
Available: <lista kart, które gracz jeszcze ma na ręce>
```

W przypadku komunikatu **TRICK** użytkownik wybiera kartę do dołożenia, wpisując wykrzyknik i jej kod, np. "!10C", i naciskając enter. Ponadto użytkownik ma do dyspozycji takie polecenia, kończące się enterem:

- **cards** – wyświetlenie listy kart na ręce;
- **tricks** – wyświetlenie listy lew wziętych w ostatniej rozgrywce w kolejności wzięcia – każda lewa to lista kart w osobnej linii.

Wszystkie listy w komunikatach dla użytkownika są wypisywane rozdzielone przecinkami i spacjami.

Wymagania formalne

Można oddać tylko implementację serwera lub tylko klienta, albo obu.

Programy mają być napisane w języku C lub C++ z wykorzystaniem interfejsu gniazd (nie wolno korzystać z **boost::asio**). Można korzystać z bibliotek pomocniczych (np. **boost::program_options**), o ile są zainstalowane w LK. Programy będą kompilowane na maszynach w LK.

Należy dostarczyć plik **Makefile** lub **makefile**. Polecenie **make** powinno tworzyć programy **kierki-serwer** i **kierki-klient**. Wśród parametrów kompilatora należy użyć **-Wall**, **-Wextra** i **-O2**, zalecamy korzystanie ze standardu **-std=gnu17** lub **-std=c++20** (w zakresie wspieranym przez kompilator w LK). Polecenie **make clean** powinno usuwać wszystkie pliki powstałe podczas kompilowania.

Jako rozwiązanie należy dostarczyć archiwum **ab123456.tgz** stworzone parą programów **tar** i **gzip**, zawierające pliki źródłowe oraz plik **makefile** lub **Makefile**, gdzie **ab123456** to standardowy login osoby oddającej rozwiązanie, używany na maszynach wydziału, wg schematu: inicjały, nr indeksu. Nie wolno umieszczać w archiwum plików zbędnych, binarnych ani pośrednich powstających podczas kompilowania.

Ocena rozwiązania

Za serwer: 12 pkt

Za klienta: 6 pkt