Laboratory Exercise

You have created *n* children from a parent and to distribute tasks to those children in **Lab 3** for a suggestion on whether to replace a card or not. In real applications, the parent *may not know* the data before-hand, until provided by the user. It is also possible that the parent would need to decide what data to be passed to which child *on demand*, e.g. pass the next task to the child which just completes its current task in the presence of many tasks to be assigned. It is thus important that a parent is able to *pass data to children **after*** they are created, similar to **lab5C.c**, which makes use of the **pipe**. More importantly, children often need to *convey results back* to the parent, since children are created by a parent usually with a goal to help handling some of the workload of the parent. The use of *exit status* could only return a single value between 0 to 255, and only when a child process terminates. Passing data back to parent when the child processes are still executing is therefore needed.

You are to extend the exercise in **Lab 3** with **lab5C.c** to play the game in **Lab 3** with communication between parent and child processes. The parent first *creates* all the necessary *pipes* (***two pipes*** for each child, one from parent to child and the other from child to parent), and then *forks* the *child processes*. Note that each pipe is associated with two file descriptors, one for read and one for write. The parent and each child will *close* the excessive ends of the respective pipes. This is very important to help program development. They will communicate via the pipes to carry out the necessary computation. Finally, everyone closes all the pipes and the parent will *wait* for all the children to terminate at the end of the game.

You are to create *n* child processes for the players ($2 \le n \le 10$) in this game. The C program is written in such a way that it accepts inputs from the ***keyboard***, e.g., using `scanf()` or `fgets()`. We will then compile this C program and ***run*** it based on the technique of *input redirection* from a file for easy testing. Thus, *end-of-file testing* should be adopted for the inputs from the ***keyboard***. The actual input data file contains a collection of cards. Each card is of the form <suit rank>, where suit is either **S** (♠ or *spade*), **H** (♥ or *heart*), **C** (♣ or *club*), or **D** (♦ or *diamond*), and rank is drawn from the set {**A**, **K**, **Q**, **J**, **T**, **9**, **8**, **7**, **6**, **5**, **4**, **3, 2**} in that order, where **T** means **10** and **A** is the highest card. You can assume that there is no error in the input lines. Parent reads in the cards until *EOF*. The cards are dealt to the children in a *round-robin manner*.

First, each player will be dealt with 5 cards. Making use of the program in **Lab 3**, each player could determine the strength of its hand. Instead of replacing a card from the deck, each player could only exchange cards among themselves. To make the game under control, the rule of exchanging card is done in an orderly manner. In each round of exchange, each player may decide to either exchange a card of its choice, or not to exchange any card. Then each player will pass its card to the next player down the road who decides to exchange a card, and receive a card from the previous player, skipping those players who decide not to exchange any card. If there are *n* players, there will be *n* exchange rounds. In a worst case, a player may get back its original card circulating through the whole table. Unlike the complex mechanism in **Lab 3**, a simplification is made here. Each player will just select the card that will not diminish the strength of its hand for exchange. In case multiple cards can be sacrificed without diminishing the strength, the lowest card is selected. Out of the nine possible types of hands in decreasing strength, `royal-flush`, `four-of-a-kind`, `full-house`, `flush`, `straight`, `three-of-a-kind`, `two-pairs`, `pair` and `nothing` (none of the eight types), it is not hard to see that a player would not choose to exchange if it is holding a hand of `royal-flush`, `full-house`, `flush` or `straight`, but would much likely decide to exchange in case of `three-of-a-kind`, `two-pairs`, `pair` or `nothing`. It is obvious that under `two-pairs`, the card for exchange would be the lone card not belonging to the two pairs. Under `three-of-a-kind` and `pair`, the card for exchange would be the lower among the two or three cards not belonging to the three or the pair. Finally, for `nothing`, simply select the lowest card for exchange. As for `four-of-a-kind`, it does not really matter whether to exchange or not. However, for simplicity, no exchange would be made.

In this program, the parent will act like the table for holding the cards, as well as the *arbitrator* to relay requests between child processes. In each round of the exchange, the parent will ask each child in turn for the card that a child would like to exchange. A child will either send this card to the parent, or reply

the parent that it decides not to exchange, via a *pipe* to the parent. After collecting the intentions from all children, the parent will pass on the card as needed to the next child in the circle, skipping those children not intended for an exchange. The parent actually serves as the *overall controller* of the whole program, prompting children for requests and passing cards to children.

Here is a simple arrangement: each child always tries to read from the pipe about the request from the parent. The parent starts the game by "writing" `yourcard` to the children by dealing them the cards in a round-robin manner. After dealing the card, in each round, the parent will "write" to each child in turn `yourchoice`, then "reads" the card or decision from that child. A child "reads" from parent and if that is `yourchoice`, it "writes" its card or a `pass` message to the parent. Parent "reads" those cards or `pass` from the children and "writes" `yourcard` to each next child in turn. After all the exchange rounds, each child reports its hand back to the parent. Finally, when all children have completed, parent will conclude the hands and announce the winner.

Do not forget to ***wait*** for all the children to complete and ***close*** the pipes in the end of the game. Please provide proper comments and check your program before submission. Your program must run on **apollo** or **apollo2**.

Sample executions (input data are stored inside a text file, e.g., **`card.txt`**, to facilitate I/O redirection):

`playcard 4 < card.txt`

```
CA CQ D3 SK C3 S2 S4 DK CJ H2   DQ SQ S3 S5 C2 D4 DA C9 ST H4   D7 S8 D6 SJ HQ D9 H9 CT C4 S6
   H7 C5 HT C8 HK D5 C6 H6 H3 C7 H8 DT HJ D2 D8 HA H5 SA S9 CK        S7   DJ
```

Sample output 1 (output lines may not be in this particular order due to concurrent process execution):

```
Parent, pid 12341: there are 4 children
Child 1, pid 12342: hand <CA C3 CJ S3 DA>
Child 2, pid 12343: hand <CQ S2 H2 S5 C9>
Child 3, pid 12344: hand <D3 S4 DQ C2 ST>
Child 4, pid 12345: hand <SK DK SQ D4 H4>
Parent, pid 12341: round 1 asking
Child 1, pid 12342: type is two-pairs, exchange card CJ
Child 2, pid 12343: type is pair, exchange card S5
Child 3, pid 12344: type is nothing, exchange card C2
Child 4, pid 12345: type is two-pairs, exchange card SQ
Parent, pid 12341: round 1 sending
Child 1, pid 12342: new card SQ, new hand <CA C3 SQ S3 DA>
Child 2, pid 12343: new card CJ, new hand <CQ S2 H2 CJ C9>
Child 3, pid 12344: new card S5, new hand <D3 S4 DQ S5 ST>
Child 4, pid 12345: new card C2, new hand <SK DK C2 D4 H4>
Parent, pid 12341: round 2 asking
Child 1, pid 12342: type is two-pairs, exchange card SQ
Child 2, pid 12343: type is pair, exchange card C9
Child 3, pid 12344: type is nothing, exchange card D3
Child 4, pid 12345: type is two-pairs, exchange card C2
Parent, pid 12341: round 2 sending
Child 1, pid 12342: new card C2, new hand <CA C3 C2 S3 DA>
Child 2, pid 12343: new card SQ, new hand <CQ S2 H2 CJ SQ>
Child 3, pid 12344: new card C9, new hand <C9 S4 DQ S5 ST>
Child 4, pid 12345: new card D3, new hand <SK DK D3 D4 H4>
Parent, pid 12341: round 3 asking
Child 1, pid 12342: type is two-pairs, exchange card C2
Child 2, pid 12343: type is two-pairs, exchange card CJ
Child 3, pid 12344: type is nothing, exchange card S4
Child 4, pid 12345: type is two-pairs, exchange card D3
Parent, pid 12341: round 3 sending
Child 1, pid 12342: new card D3, new hand <CA C3 D3 S3 DA>
Child 2, pid 12343: new card C2, new hand <CQ S2 H2 C2 SQ>
Child 3, pid 12344: new card CJ, new hand <C9 CJ DQ S5 ST>
Child 4, pid 12345: new card S4, new hand <SK DK S4 D4 H4>
Parent, pid 12341: round 4 asking
Child 1, pid 12342: type is full-house, no exchange
Child 2, pid 12343: type is full-house, no exchange
Child 3, pid 12344: type is nothing, exchange card S5
Child 4, pid 12345: type is full-house, no exchange
Parent, pid 12341: round 4 sending
Child 3, pid 12344: new card S5, new hand <C9 CJ DQ S5 ST>
Child 1, pid 12342: improve from two-pairs to full-house
Child 2, pid 12343: improve from pair to full-house
Child 3, pid 12344: no improvement for nothing
```

```
Child 4, pid 12345: improve from two-pairs to full-house
Parent, pid 12341: child 1 hand <CA C3 D3 S3 DA> is full-house
Parent, pid 12341: child 2 hand <CQ S2 H2 C2 SQ> is full-house
Parent, pid 12341: child 3 hand <C9 CJ DQ S5 ST> is nothing
Parent, pid 12341: child 4 hand <SK DK S4 D4 H4> is full-house
Parent, pid 12341: child 4 is winner
```

playcard 5 < card.txt

```
CA CQ D3 SK C3 S2 S4 DK CJ H2   DQ SQ S3 S5 C2 D4 DA C9 ST H4   D7 S8 D6 SJ HQ D9 H9 CT C4 S6
   H7 C5 HT C8 HK D5 C6 H6 H3 C7 H8 DT HJ D2 D8 HA H5 SA S9 CK        S7   DJ
```

Sample output 2:

```
Parent, pid 12351: there are 5 children
Child 1, pid 12352: hand <CA S2 DQ D4 D7>
Child 2, pid 12353: hand <CQ S4 SQ DA S8>
Child 3, pid 12354: hand <D3 DK S3 C9 D6>
Child 4, pid 12355: hand <SK CJ S5 ST SJ>
Child 5, pid 12356: hand <C3 H2 C2 H4 HQ>
Parent, pid 12351: round 1 asking
Child 1, pid 12352: type is nothing, exchange card S2
Child 2, pid 12353: type is pair, exchange card S4
Child 3, pid 12354: type is pair, exchange card D6
Child 4, pid 12355: type is pair, exchange card S5
Child 5, pid 12356: type is pair, exchange card C3
Parent, pid 12351: round 1 sending
Child 1, pid 12352: new card C3, new hand <CA C3 DQ D4 D7>
Child 2, pid 12353: new card S2, new hand <CQ S2 SQ DA S8>
Child 3, pid 12354: new card S4, new hand <D3 DK S3 C9 S4>
Child 4, pid 12355: new card D6, new hand <SK CJ D6 ST SJ>
Child 5, pid 12356: new card S5, new hand <S5 H2 C2 H4 HQ>
Parent, pid 12351: round 2 asking
Child 1, pid 12352: type is nothing, exchange card C3
Child 2, pid 12353: type is pair, exchange card S2
Child 3, pid 12354: type is pair, exchange card S4
Child 4, pid 12355: type is pair, exchange card D6
Child 5, pid 12356: type is pair, exchange card H4
Parent, pid 12351: round 2 sending
Child 1, pid 12352: new card H4, new hand <CA H4 DQ D4 D7>
Child 2, pid 12353: new card C3, new hand <CQ C3 SQ DA S8>
Child 3, pid 12354: new card S2, new hand <D3 DK S3 C9 S2>
Child 4, pid 12355: new card S4, new hand <SK CJ S4 ST SJ>
Child 5, pid 12356: new card D6, new hand <S5 H2 C2 D6 HQ>
Parent, pid 12351: round 3 asking
Child 1, pid 12352: type is pair, exchange card D7
Child 2, pid 12353: type is pair, exchange card C3
Child 3, pid 12354: type is pair, exchange card S2
Child 4, pid 12355: type is pair, exchange card S4
Child 5, pid 12356: type is pair, exchange card S5
Parent, pid 12351: round 3 sending
Child 1, pid 12352: new card S5, new hand <CA H4 DQ D4 S5>
Child 2, pid 12353: new card D7, new hand <CQ D7 SQ DA S8>
Child 3, pid 12354: new card C3, new hand <D3 DK S3 C9 C3>
Child 4, pid 12355: new card S2, new hand <SK CJ S2 ST SJ>
Child 5, pid 12356: new card S4, new hand <S4 H2 C2 D6 HQ>
Parent, pid 12351: round 4 asking
Child 1, pid 12352: type is pair, exchange card S5
Child 2, pid 12353: type is pair, exchange card D7
Child 3, pid 12354: type is three-of-a-kind, exchange card C9
Child 4, pid 12355: type is pair, exchange card S2
Child 5, pid 12356: type is pair, exchange card S4
Parent, pid 12351: round 4 sending
Child 1, pid 12352: new card S4, new hand <CA H4 DQ D4 S4>
Child 2, pid 12353: new card S5, new hand <CQ S5 SQ DA S8>
Child 3, pid 12354: new card D7, new hand <D3 DK S3 D7 C3>
Child 4, pid 12355: new card C9, new hand <SK CJ C9 ST SJ>
Child 5, pid 12356: new card S2, new hand <S2 H2 C2 D6 HQ>
Parent, pid 12351: round 5 asking
Child 1, pid 12352: type is three-of-a-kind, exchange card DQ
Child 2, pid 12353: type is pair, exchange card S5
Child 3, pid 12354: type is three-of-a-kind, exchange card D7
Child 4, pid 12355: type is pair, exchange card C9
Child 5, pid 12356: type is three-of-a-kind, exchange card D6
Parent, pid 12351: round 5 sending
Child 1, pid 12352: new card D6, new hand <CA H4 D6 D4 S4>
Child 2, pid 12353: new card DQ, new hand <CQ DQ SQ DA S8>
```

```
Child 3, pid 12354: new card S5, new hand <D3 DK S3 S5 C3>
Child 4, pid 12355: new card D7, new hand <SK CJ D7 ST SJ>
Child 5, pid 12356: new card C9, new hand <S2 H2 C2 C9 HQ>
Child 1, pid 12352: improve from nothing to three-of-a-kind
Child 2, pid 12353: improve from pair to three-of-a-kind
Child 3, pid 12354: improve from pair to three-of-a-kind
Child 4, pid 12355: no improvement for pair
Child 5, pid 12356: improve from pair to three-of-a-kind
Parent, pid 12351: child 1 hand <CA H4 D6 D4 S4> is three-of-a-kind
Parent, pid 12351: child 2 hand <CQ DQ SQ DA S8> is three-of-a-kind
Parent, pid 12351: child 3 hand <D3 DK S3 S5 C3> is three-of-a-kind
Parent, pid 12351: child 4 hand <SK CJ D7 ST SJ> is pair
Parent, pid 12351: child 5 hand <S2 H2 C2 C9 HQ> is three-of-a-kind
Parent, pid 12351: child 2 is winner
```

playcard 6 < card.txt

```
CA CQ D3 SK C3 S2 S4 DK CJ H2   DQ SQ S3 S5 C2 D4 DA C9 ST H4   D7 S8 D6 SJ HQ D9 H9 CT C4 S6
   H7 C5 HT C8 HK D5 C6 H6 H3 C7 H8 DT HJ D2 D8 HA H5 SA S9 CK        S7   DJ
```

Sample output 3:

```
Parent, pid 12361: there are 6 children
Child 1, pid 12362: hand <CA S4 S3 ST HQ>
Child 2, pid 12363: hand <CQ DK S5 H4 D9>
Child 3, pid 12364: hand <D3 CJ C2 D7 H9>
Child 4, pid 12365: hand <SK H2 D4 S8 CT>
Child 5, pid 12366: hand <C3 DQ DA D6 C4>
Child 6, pid 12367: hand <S2 SQ C9 SJ S6>
Parent, pid 12361: round 1 asking
Child 1, pid 12362: type is nothing, exchange card S3
Child 2, pid 12363: type is nothing, exchange card H4
Child 3, pid 12364: type is nothing, exchange card C2
Child 4, pid 12365: type is nothing, exchange card H2
Child 5, pid 12366: type is nothing, exchange card C3
Child 6, pid 12367: type is nothing, exchange card S2
Parent, pid 12361: round 1 sending
Child 1, pid 12362: new card S2, new hand <CA S4 S2 ST HQ>
Child 2, pid 12363: new card S3, new hand <CQ DK S5 S3 D9>
Child 3, pid 12364: new card H4, new hand <D3 CJ H4 D7 H9>
Child 4, pid 12365: new card C2, new hand <SK C2 D4 S8 CT>
Child 5, pid 12366: new card H2, new hand <H2 DQ DA D6 C4>
Child 6, pid 12367: new card C3, new hand <C3 SQ C9 SJ S6>
Parent, pid 12361: round 2 asking
Child 1, pid 12362: type is nothing, exchange card S2
Child 2, pid 12363: type is nothing, exchange card S3
Child 3, pid 12364: type is nothing, exchange card D3
Child 4, pid 12365: type is nothing, exchange card C2
Child 5, pid 12366: type is nothing, exchange card H2
Child 6, pid 12367: type is nothing, exchange card C3
Parent, pid 12361: round 2 sending
Child 1, pid 12362: new card C3, new hand <CA S4 C3 ST HQ>
Child 2, pid 12363: new card S2, new hand <CQ DK S5 S2 D9>
Child 3, pid 12364: new card S3, new hand <S3 CJ H4 D7 H9>
Child 4, pid 12365: new card D3, new hand <SK D3 D4 S8 CT>
Child 5, pid 12366: new card C2, new hand <C2 DQ DA D6 C4>
Child 6, pid 12367: new card H2, new hand <H2 SQ C9 SJ S6>
Parent, pid 12361: round 3 asking
Child 1, pid 12362: type is nothing, exchange card C3
Child 2, pid 12363: type is nothing, exchange card S2
Child 3, pid 12364: type is nothing, exchange card S3
Child 4, pid 12365: type is nothing, exchange card D3
Child 5, pid 12366: type is nothing, exchange card C2
Child 6, pid 12367: type is nothing, exchange card H2
Parent, pid 12361: round 3 sending
Child 1, pid 12362: new card H2, new hand <CA S4 H2 ST HQ>
Child 2, pid 12363: new card C3, new hand <CQ DK S5 C3 D9>
Child 3, pid 12364: new card S2, new hand <S2 CJ H4 D7 H9>
Child 4, pid 12365: new card S3, new hand <SK S3 D4 S8 CT>
Child 5, pid 12366: new card D3, new hand <D3 DQ DA D6 C4>
Child 6, pid 12367: new card C2, new hand <C2 SQ C9 SJ S6>
Parent, pid 12361: round 4 asking
Child 1, pid 12362: type is nothing, exchange card H2
Child 2, pid 12363: type is nothing, exchange card C3
Child 3, pid 12364: type is nothing, exchange card S2
Child 4, pid 12365: type is nothing, exchange card S3
Child 5, pid 12366: type is nothing, exchange card D3
```

```
Child 6, pid 12367: type is nothing, exchange card C2
Parent, pid 12361: round 4 sending
Child 1, pid 12362: new card C2, new hand <CA S4 C2 ST HQ>
Child 2, pid 12363: new card H2, new hand <CQ DK S5 H2 D9>
Child 3, pid 12364: new card C3, new hand <C3 CJ H4 D7 H9>
Child 4, pid 12365: new card S2, new hand <SK S2 D4 S8 CT>
Child 5, pid 12366: new card S3, new hand <S3 DQ DA D6 C4>
Child 6, pid 12367: new card D3, new hand <D3 SQ C9 SJ S6>
Parent, pid 12361: round 5 asking
Child 1, pid 12362: type is nothing, exchange card C2
Child 2, pid 12363: type is nothing, exchange card H2
Child 3, pid 12364: type is nothing, exchange card C3
Child 4, pid 12365: type is nothing, exchange card S2
Child 5, pid 12366: type is nothing, exchange card S3
Child 6, pid 12367: type is nothing, exchange card D3
Parent, pid 12361: round 5 sending
Child 1, pid 12362: new card D3, new hand <CA S4 D3 ST HQ>
Child 2, pid 12363: new card C2, new hand <CQ DK S5 C2 D9>
Child 3, pid 12364: new card H2, new hand <H2 CJ H4 D7 H9>
Child 4, pid 12365: new card C3, new hand <SK C3 D4 S8 CT>
Child 5, pid 12366: new card S2, new hand <S2 DQ DA D6 C4>
Child 6, pid 12367: new card S3, new hand <S3 SQ C9 SJ S6>
Parent, pid 12361: round 6 asking
Child 1, pid 12362: type is nothing, exchange card D3
Child 2, pid 12363: type is nothing, exchange card C2
Child 3, pid 12364: type is nothing, exchange card H2
Child 4, pid 12365: type is nothing, exchange card C3
Child 5, pid 12366: type is nothing, exchange card S2
Child 6, pid 12367: type is nothing, exchange card S3
Parent, pid 12361: round 6 sending
Child 1, pid 12362: new card S3, new hand <CA S4 S3 ST HQ>
Child 2, pid 12363: new card D3, new hand <CQ DK S5 D3 D9>
Child 3, pid 12364: new card C2, new hand <C2 CJ H4 D7 H9>
Child 4, pid 12365: new card H2, new hand <SK H2 D4 S8 CT>
Child 5, pid 12366: new card C3, new hand <C3 DQ DA D6 C4>
Child 6, pid 12367: new card S2, new hand <S2 SQ C9 SJ S6>
Child 1, pid 12362: no improvement for nothing
Child 2, pid 12363: no improvement for nothing
Child 3, pid 12364: no improvement for nothing
Child 4, pid 12365: no improvement for nothing
Child 5, pid 12366: no improvement for nothing
Child 6, pid 12367: no improvement for nothing
Parent, pid 12361: child 1 hand <CA S4 S3 ST HQ> is nothing
Parent, pid 12361: child 2 hand <CQ DK S5 D3 D9> is nothing
Parent, pid 12361: child 3 hand <C2 CJ H4 D7 H9> is nothing
Parent, pid 12361: child 4 hand <SK H2 D4 S8 CT> is nothing
Parent, pid 12361: child 5 hand <C3 DQ DA D6 C4> is nothing
Parent, pid 12361: child 6 hand <S2 SQ C9 SJ S6> is nothing
Parent, pid 12361: child 1 is winner
```

playcard 2 < card.txt

```
C4 HT HK C5 D6 S5 S9 C6 H3 HJ D4 ST DT CK CJ HA D9 C9 D7 D5 D2 SA C8 SJ S6 SK H5 H2 S4 DQ DK C2
D8 H6 SQ S7 S3 CT H9 S2 CA H4 DA C7 H8 HQ CQ C3 D3 S8 DJ H7
```

Sample output 4:

```
Parent, pid 12411: there are 2 children
Child 1, pid 12412: hand <C4 HK D6 S9 H3>
Child 2, pid 12413: hand <HT C5 S5 C6 HJ>
Parent, pid 12411: round 1 asking
Child 1, pid 12412: type is nothing, exchange card H3
Child 2, pid 12413: type is pair, exchange card C6
Parent, pid 12411: round 1 sending
Child 1, pid 12412: new card C6, new hand <C4 HK D6 S9 C6>
Child 2, pid 12413: new card H3, new hand <HT C5 S5 H3 HJ>
Parent, pid 12411: round 2 asking
Child 1, pid 12412: type is pair, exchange card C4
Child 2, pid 12413: type is pair, exchange card H3
Parent, pid 12411: round 2 sending
Child 1, pid 12412: new card H3, new hand <H3 HK D6 S9 C6>
Child 2, pid 12413: new card C4, new hand <HT C5 S5 C4 HJ>
Child 1, pid 12412: improve from nothing to pair
Child 2, pid 12413: no improvement for pair
Parent, pid 12411: child 1 hand <H3 HK D6 S9 C6> is pair
Parent, pid 12411: child 2 hand <HT C5 S5 C4 HJ> is pair
Parent, pid 12411: child 1 is winner
```

playcard 5 < card.txt

```
C4 HT HK C5 D6 S5 S9 C6 H3 HJ D4 ST DT CK CJ HA D9 C9 D7 D5 D2 SA C8 SJ S6 SK H5 H2 S4 DQ DK C2
D8 H6 SQ S7 S3 CT H9 S2 CA H4 DA C7 H8 HQ CQ C3 D3 S8 DJ H7
```

Sample output 5:

```
Parent, pid 12421: there are 5 children
Child 1, pid 12422: hand <C4 S5 D4 HA D2>
Child 2, pid 12423: hand <HT S9 ST D9 SA>
Child 3, pid 12424: hand <HK C6 DT C9 C8>
Child 4, pid 12425: hand <C5 H3 CK D7 SJ>
Child 5, pid 12426: hand <D6 HJ CJ D5 S6>
Parent, pid 12421: round 1 asking
Child 1, pid 12422: type is pair, exchange card D2
Child 2, pid 12423: type is two-pairs, exchange card SA
Child 3, pid 12424: type is nothing, exchange card C6
Child 4, pid 12425: type is nothing, exchange card H3
Child 5, pid 12426: type is two-pairs, exchange card D5
Parent, pid 12421: round 1 sending
Child 1, pid 12422: new card D5, new hand <C4 S5 D4 HA D5>
Child 2, pid 12423: new card D2, new hand <HT S9 ST D9 D2>
Child 3, pid 12424: new card SA, new hand <HK SA DT C9 C8>
Child 4, pid 12425: new card C6, new hand <C5 C6 CK D7 SJ>
Child 5, pid 12426: new card H3, new hand <D6 HJ CJ H3 S6>
. . .
Parent, pid 12421: round 5 sending
Child 2, pid 12423: new card D7, new hand <HT S9 ST D9 D7>
Child 3, pid 12424: new card D2, new hand <HK SA DT D2 HA>
Child 4, pid 12425: new card H3, new hand <C8 C9 CK H3 SJ>
Child 1, pid 12422: improve from pair to full-house
Child 2, pid 12423: no improvement for two-pairs
Child 3, pid 12424: improve from nothing to pair
Child 4, pid 12425: no improvement for nothing
Child 5, pid 12426: improve from two-pairs to full-house
Parent, pid 12421: child 1 hand <C4 S5 D4 C5 D5> is full-house
Parent, pid 12421: child 2 hand <HT S9 ST D9 D7> is two-pairs
Parent, pid 12421: child 3 hand <HK SA DT D2 HA> is pair
Parent, pid 12421: child 4 hand <C8 C9 CK H3 SJ> is nothing
Parent, pid 12421: child 5 hand <D6 HJ CJ C6 S6> is full-house
Parent, pid 12421: child 5 is winner
```

playcard 7 < card.txt

```
DK DQ S4 S8 H3 CJ C2 D3 HA SK S2 CT H8 ST H7 HT HJ C4 C8 D7 H4 C3 CQ C6 D8 H9 D5 DJ S5 HQ D2 S7
C5 HK H6 H2 S6 D6 SA SQ C9 D9 DA H5 S3 CA D4 C7 CK S9 DT SJ
```

Sample output 6:

```
Parent, pid 12521: there are 7 children
Child 1, pid 12522: hand <DK D3 H7 C3 S5>
Child 2, pid 12523: hand <DQ HA HT CQ HQ>
Child 3, pid 12524: hand <S4 SK HJ C6 D2>
Child 4, pid 12525: hand <S8 S2 C4 D8 S7>
Child 5, pid 12526: hand <H3 CT C8 H9 C5>
Child 6, pid 12527: hand <CJ H8 D7 D5 HK>
Child 7, pid 12528: hand <C2 ST H4 DJ H6>
Parent, pid 12521: round 1 asking
Child 1, pid 12522: type is pair, exchange card S5
Child 2, pid 12523: type is three-of-a-kind, exchange card HT
Child 3, pid 12524: type is nothing, exchange card D2
Child 4, pid 12525: type is pair, exchange card S2
Child 5, pid 12526: type is nothing, exchange card H3
Child 6, pid 12527: type is nothing, exchange card D5
Child 7, pid 12528: type is nothing, exchange card C2
Parent, pid 12521: round 1 sending
Child 1, pid 12522: new card C2, new hand <DK D3 H7 C3 C2>
Child 2, pid 12523: new card S5, new hand <DQ HA S5 CQ HQ>
Child 3, pid 12524: new card HT, new hand <S4 SK HJ C6 HT>
Child 4, pid 12525: new card D2, new hand <S8 D2 C4 D8 S7>
Child 5, pid 12526: new card S2, new hand <S2 CT C8 H9 C5>
Child 6, pid 12527: new card H3, new hand <CJ H8 D7 H3 HK>
Child 7, pid 12528: new card D5, new hand <D5 ST H4 DJ H6>
. . .
Parent, pid 12521: round 7 sending
Child 1, pid 12522: new card ST, new hand <DK D3 ST C3 H3>
```

```
Child 2, pid 12523: new card H6, new hand <DQ HA H6 CQ HQ>
Child 3, pid 12524: new card D2, new hand <H7 SK HJ D2 HT>
Child 6, pid 12527: new card S2, new hand <CJ H8 D7 S2 HK>
Child 7, pid 12528: new card C2, new hand <D5 C2 C5 DJ S5>
Child 1, pid 12522: improve from pair to three-of-a-kind
Child 2, pid 12523: no improvement for three-of-a-kind
Child 3, pid 12524: no improvement for nothing
Child 4, pid 12525: improve from pair to full-house
Child 5, pid 12526: improve from nothing to straight
Child 6, pid 12527: no improvement for nothing
Child 7, pid 12528: improve from nothing to three-of-a-kind
Parent, pid 12521: child 1 hand <DK D3 ST C3 H3> is three-of-a-kind
Parent, pid 12521: child 2 hand <DQ HA H6 CQ HQ> is three-of-a-kind
Parent, pid 12521: child 3 hand <H7 SK HJ D2 HT> is nothing
Parent, pid 12521: child 4 hand <S8 S4 C4 D8 H4> is full-house
Parent, pid 12521: child 5 hand <S7 CT C8 H9 C6> is straight
Parent, pid 12521: child 6 hand <CJ H8 D7 S2 HK> is nothing
Parent, pid 12521: child 7 hand <D5 C2 C5 DJ S5> is three-of-a-kind
Parent, pid 12521: child 4 is winner
```

playcard 8 < card.txt

```
DK DQ S4 S8 H3 CJ C2 D3 HA SK S2 CT H8 ST H7 HT HJ C4 C8 D7 H4 C3 CQ C6 D8 H9 D5 DJ S5 HQ D2 S7
C5 HK H6 H2 S6 D6 SA SQ C9 D9 DA H5 S3 CA D4 C7 CK S9 DT SJ
```

Sample output 7:

```
Parent, pid 12531: there are 8 children
Child 1, pid 12532: hand <DK HA HJ D8 C5>
Child 2, pid 12533: hand <DQ SK C4 H9 HK>
Child 3, pid 12534: hand <S4 S2 C8 D5 H6>
Child 4, pid 12535: hand <S8 CT D7 DJ H2>
Child 5, pid 12536: hand <H3 H8 H4 S5 S6>
Child 6, pid 12537: hand <CJ ST C3 HQ D6>
Child 7, pid 12538: hand <C2 H7 CQ D2 SA>
Child 8, pid 12539: hand <D3 HT C6 S7 SQ>
Parent, pid 12531: round 1 asking
Child 1, pid 12532: type is nothing, exchange card C5
Child 2, pid 12533: type is pair, exchange card C4
Child 3, pid 12534: type is nothing, exchange card S2
Child 4, pid 12535: type is nothing, exchange card H2
Child 5, pid 12536: type is nothing, exchange card H3
Child 6, pid 12537: type is nothing, exchange card C3
Child 7, pid 12538: type is pair, exchange card H7
Child 8, pid 12539: type is nothing, exchange card D3
. . .
Parent, pid 12531: round 8 sending
Child 1, pid 12532: new card D3, new hand <DK HA D3 ST HT>
Child 2, pid 12533: new card HJ, new hand <DQ SK HJ H9 HK>
Child 3, pid 12534: new card H4, new hand <S4 C4 H4 C6 H6>
Child 4, pid 12535: new card D8, new hand <S8 CT D8 DJ C8>
Child 5, pid 12536: new card H3, new hand <D5 H8 C5 S5 H3>
Child 6, pid 12537: new card C3, new hand <CJ C3 S6 HQ D6>
Child 8, pid 12539: new card D7, new hand <H7 D7 CQ S7 SQ>
Child 1, pid 12532: improve from nothing to pair
Child 2, pid 12533: no improvement for pair
Child 3, pid 12534: improve from nothing to full-house
Child 4, pid 12535: improve from nothing to three-of-a-kind
Child 5, pid 12536: improve from nothing to three-of-a-kind
Child 6, pid 12537: improve from nothing to pair
Child 7, pid 12538: improve from pair to four-of-a-kind
Child 8, pid 12539: improve from nothing to full-house
Parent, pid 12531: child 1 hand <DK HA D3 ST HT> is pair
Parent, pid 12531: child 2 hand <DQ SK HJ H9 HK> is pair
Parent, pid 12531: child 3 hand <S4 C4 H4 C6 H6> is full-house
Parent, pid 12531: child 4 hand <S8 CT D8 DJ C8> is three-of-a-kind
Parent, pid 12531: child 5 hand <D5 H8 C5 S5 H3> is three-of-a-kind
Parent, pid 12531: child 6 hand <CJ C3 S6 HQ D6> is pair
Parent, pid 12531: child 7 hand <C2 H2 S2 D2 SA> is four-of-a-kind
Parent, pid 12531: child 8 hand <H7 D7 CQ S7 SQ> is full-house
Parent, pid 12531: child 7 is winner
```

playcard 10 < card.txt

```
DK DQ S4 S8 H3 CJ C2 D3 HA SK S2 CT H8 ST H7 HT HJ C4 C8 D7 H4 C3 CQ C6 D8 H9 D5 DJ S5 HQ D2 S7
C5 HK H6 H2 S6 D6 SA SQ C9 D9 DA H5 S3 CA D4 C7 CK S9 DT SJ
```

Sample output 8:

```
Parent, pid 12541: there are 10 children
Child 1, pid 12542: hand <DK S2 H4 D2 C9>
Child 2, pid 12543: hand <DQ CT C3 S7 D9>
Child 3, pid 12544: hand <S4 H8 CQ C5 DA>
Child 4, pid 12545: hand <S8 ST C6 HK H5>
Child 5, pid 12546: hand <H3 H7 D8 H6 S3>
Child 6, pid 12547: hand <CJ HT H9 H2 CA>
Child 7, pid 12548: hand <C2 HJ D5 S6 D4>
Child 8, pid 12549: hand <D3 C4 DJ D6 C7>
Child 9, pid 12550: hand <HA C8 S5 SA CK>
Child 10, pid 12551: hand <SK D7 HQ SQ S9>
Parent, pid 12541: round 1 asking
Child 1, pid 12542: type is pair, exchange card H4
Child 2, pid 12543: type is nothing, exchange card C3
Child 3, pid 12544: type is nothing, exchange card S4
Child 4, pid 12545: type is nothing, exchange card H5
Child 5, pid 12546: type is pair, exchange card H6
Child 6, pid 12547: type is nothing, exchange card H2
Child 7, pid 12548: type is nothing, exchange card C2
Child 8, pid 12549: type is nothing, exchange card D3
Child 9, pid 12550: type is pair, exchange card S5
Child 10, pid 12551: type is pair, exchange card D7
. . .
Child 1, pid 12542: improve from pair to four-of-a-kind
Child 2, pid 12543: improve from nothing to four-of-a-kind
Child 3, pid 12544: improve from nothing to pair
Child 4, pid 12545: improve from nothing to two-pairs
Child 5, pid 12546: improve from pair to four-of-a-kind
Child 6, pid 12547: no improvement for nothing
Child 7, pid 12548: improve from nothing to full-house
Child 8, pid 12549: improve from nothing to full-house
Child 9, pid 12550: no improvement for pair
Child 10, pid 12551: no improvement for pair
Parent, pid 12541: child 1 hand <DK S2 C2 D2 H2> is four-of-a-kind
Parent, pid 12541: child 2 hand <DQ H7 D7 S7 C7> is four-of-a-kind
Parent, pid 12541: child 3 hand <D9 S5 CQ C9 DA> is pair
Parent, pid 12541: child 4 hand <S8 ST CT H4 H8> is two-pairs
Parent, pid 12541: child 5 hand <H3 D3 D8 C3 S3> is four-of-a-kind
Parent, pid 12541: child 6 hand <CJ HT H9 HK CA> is nothing
Parent, pid 12541: child 7 hand <H6 C5 D5 S6 H5> is full-house
Parent, pid 12541: child 8 hand <D4 C4 DJ HJ S4> is full-house
Parent, pid 12541: child 9 hand <HA C8 D6 SA CK> is pair
Parent, pid 12541: child 10 hand <SK C6 HQ SQ S9> is pair
Parent, pid 12541: child 2 is winner
```

Level 1 requirement: *n* child processes are created and capable of printing out the cards received (this is similar to level 1 in **Lab 3 exercise**), ***and*** I/O redirection is used.
Level 2 requirement: the child processes can communicate with the parent and the game plays correctly for part of the first round.
Level 3 requirement: the game is quite often played correctly.
Level 4 requirement: the game is played correctly, and the parent could mostly announce the winner.
Bonus level: Better strategies for exchange are devised. For example, if a `nothing` contains already four cards of the same suit, it will be far better to exchange for the card of the other suit in an attempt for a `flush`. Sometimes, one may take the risk by breaking up a `pair` in order to try for a `straight` or a `flush`. Explain your implemented strategy together with a test data file to demonstrate your program.

Name your program **playcard.c** and ***submit it via BlackBoard on or before 24 March, 2021***.