```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>
#include<math.h>
#include<time.h>
#include<conio.h>

/* Differences:
Queue = (FIFO) First in First out = pushHead(), popTail();
Stack = (LIFO) Last in First out = pushHead(), popHead();

*/

void PauseEnter(){
    printf("\n[ENTER] Continue");
    getchar(); getchar(); //Visual Studio 2017 is broken, igonred first getchar! Please use Eclipse instead next time!
}

void RinseScreen(){
    system("cls");
    for(int i = 0; i < 25; i++){
        printf("\n");
    }
}

struct Data{ //Stack it!
    int tipe;
    float weight;

    Data *next;
    Data *prev;
} *head = NULL, *tail = NULL, *nd = NULL;


void pushHead(int tipe, float weight){
    nd = (Data *) malloc(sizeof(Data));

    nd->tipe = tipe;
    nd->weight = (float) weight;

    if(head==NULL){
        head=tail=nd;
    } else {
        head->prev = nd; //Forward arrow
        nd->next = head; //Backward arrow, remember to connect back to it's predecessor
        head = nd;
    }
    tail->next = NULL;
    head->prev = NULL;
}

void pushTail(int tipe, float weight){
    nd = (Data *) malloc(sizeof(Data));
```

```
55
56      nd->tipe = tipe;
57      nd->weight = (float) weight;
58
59      if(head==NULL){
60          head=tail=nd;
61      } else {
62          tail->next = nd; //Forward arrow
63          nd->prev = tail; //Backward arrow, remember to connect back to it's
              predecessor
64          tail = nd;
65      }
66      tail->next = NULL;
67      head->prev = NULL;
68  }
69
70  void popHead(){ //erase first data from head
71      if(head){ //head != NULL
72          //if there is data
73          if(head==tail){
74              //if data is only consists 1 set
75              free(head); //or tail, whatever.
76              head = tail = NULL;
77          } else {
78              head = head->next;
79              free(head->prev); //big advantage of double linked list, no need
                nd. reduces data loss
80              head->prev = NULL; //reassign deleted data as NULL
81          }
82      } else {
83          //if there is no data
84          printf("tidak ada data");
85      }
86  }
87
88  void popTail(){ //erase first data from tail
89      if(head){ //head != NULL
90          //if there is data
91          if(tail==head){
92              //if data is only consists 1 set
93              free(tail); //or head, whatever.
94              head = tail = NULL;
95          } else {
96              tail = tail->prev;
97              free(tail->next); //big advantage of double linked list, no need
                nd. reduces data loss
98              tail->next = NULL; //reassign deleted data as NULL
99          }
100     } else {
101         //if there is no data
102         printf("tidak ada data");
103     }
104 }
105
106 void popSearch(float weight){ //this one is better
107     if(head){
```

```
108         nd = head;
109 
110         while(nd!=NULL && nd->weight != (float) weight){
111             nd = nd->next;
112         }
113         if(nd==NULL){
114             printf("weight tidak ditemukan");
115         } else {
116             if(nd==head)popHead();
117             else if(nd==tail)popTail();
118             else {
119                 nd->prev->next = nd->next;
120                 nd->next->prev = nd->prev;
121                 free(nd);
122             }
123         }
124     }
125 }
126 
127 void popAll(){
128     while(head != NULL) {
129         popHead();
130     }
131 }
132 
133 void view(){
134     nd = head;
135     printf("NULL<->");
136     while(nd != NULL){
137         printf("%f<->", nd->weight);
138         nd = nd->next;
139     }
140     printf("NULL");
141 }
142 
143 void printDatas(){
144     nd = head;
145     char KenfortType[50];
146 
147     while(nd != NULL){
148         if(nd->tipe == 0){
149             strcpy(KenfortType, "long");
150         } else if (nd->tipe == 1) {
151             strcpy(KenfortType, "medium");
152         } else if (nd->tipe == 2) {
153             strcpy(KenfortType, "short");
154         } else {
155             strcpy(KenfortType, "unknown");
156         }
157         printf("[%s grain   |   %.0f kg(s)]", KenfortType, (float) nd-
              >weight);
158         if(nd == head)printf(" -> [top]");
159         printf("\n");
160         nd = nd->next;
161     }
162 }
```

```
163
164
165 void menus(int kountings){
166     RinseScreen();
167     printf("BLUE RICE STOCK\n");
168     printf("^^^^^^^^^^^^^^^\n");
169     printf("\n");
170     printf(" Rice Stock(STACK) | %d Sack(s)\n", kountings);
171     printf("\n");
172     printDatas();
173     printf("\n");
174     printf("\n");
175     printf("Option: \n");
176     printf("1. Stock Rice Sack ");
177
178     if (kountings == 10) printf("[Already FULL!!!]");
179     else if (kountings > 7 && kountings < 10) printf("[Almost full!]");
180     else if (kountings > 10) printf("[OVERLOAD!!! YOUR SHOP WILL
          EXPLODE!!!]");
181     printf("\n");
182
183     printf("2. Sell Rice Sack ");
184
185     if (kountings > 1 && kountings < 3) printf("[Going to out of stock, Low
          stock!]");
186     else if(kountings == 0) printf("[EMPTY!!! Out of Stock!!!]");
187     else if(kountings > 0) printf("[How could negative stock is possible?!]");
188     printf("\n");
189
190     printf("3. Exit\n");
191     printf("\n");
192     printf(">> Input choice: ");
193
194 }
195
196 void Stock(int & kountings){
197     RinseScreen();
198     char InsertChar[50];
199     float InsertFloat= 0;
200     int ConvertType= 0;
201
202
203     if (kountings < 10) {
204         do {
205             printf("Input Rice Type [long/medium/short]: ");
206             scanf("%s", &InsertChar); fflush(stdin);
207         } while ((strcmp(InsertChar, "long") != 0 && strcmp(InsertChar,
              "medium")) != 0 && strcmp(InsertChar, "short") != 0);
208
209         if (strcmp(InsertChar, "long") == 0) ConvertType = 0;
210         else if (strcmp(InsertChar, "medium") == 0)  ConvertType = 1;
211         else if (strcmp(InsertChar, "short") == 0) ConvertType = 2;
212         else ConvertType = -1;
213
214         do {
215             printf("Input Weight of The Rice Sack [10..100 Kg(s)]: ");
```

```
216             scanf("%f", &InsertFloat); fflush(stdin);
217         } while ((float)InsertFloat < 10 || (float)InsertFloat > 100);
218
219         printf("Stock:\n");
220         printf("[%s grain   |   %.0f kg(s)]\n", InsertChar, InsertFloat);
221         pushHead(ConvertType, InsertFloat);
222
223         kountings++;
224
225         printf("\n---Add Rice Stack Success! (Total = %d Sack(s)) ---\n",
              kountings);
226     }
227     else {
228         printf("--- The Rice Storage is Full (Total = %d Sack(s)) ---",
              kountings);
229     }
230
231     PauseEnter();
232 }
233
234 void Sell(int & kountings){
235     RinseScreen();
236
237
238
239     if(head){
240         printf("Remove Rice Sack Head");
241         popHead();
242
243         kountings--;
244
245         printf("\n--- Sell Rice Stack Success (Total = %d Sack(s)) ---\n",
              kountings);
246     } else {
247         printf("\n--- The Rice Storage is Empty (Total = %d Sack(s)) ---\n",
              kountings);
248     }
249
250     PauseEnter();
251 }
252
253
254 int main(){
255     int select;
256     int kounter = 0;
257
258     do{
259         menus(kounter);
260         scanf("%d", &select); fflush(stdin);
261
262         switch(select){
263         case 1: //Stock
264             Stock(kounter);
265             break;
266         case 2: //Sell
267             Sell(kounter);
```

```
268             break;
269         case 3: //Exit
270             break;
271         default: //error
272             break;
273         }
274
275     }while(select != 3);
276
277     return 0;
278 }
279
280 //source name: NIM_TaskName.cpp
```