7/12/2020.

1) Compliment linked list: Insert front, delete rear, display, sort, search, reverse

```
#include <stdio.h>
#include <stdlib>
struct node
{
int info;
struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
Node x;
x = (NODE)malloc(sizeof(struct node));
if (x == NULL)
{
printf("mem full\n");
exit(0)
}
return x;
}
Void freenode(NODE x)
{
free(x);
}
NODE insert_front(NODE first, int item)
{
NODE temp;
temp = getnode();
temp->info = item;
temp->link = NULL;
```

```
if (first == NULL)
  return temp;
temp->link = first;
 first=temp;
return first;
}

NODE delete_rear (NODE first)
{
NODE cur, prev;
if (first==NULL)
{
printf (" list empty can't delete \n");
return first;
}
if (first->link == NULL)
{
printf (" item deleted is %d\n", first->info);
free (first);
return NULL;
}
prev = NULL;
cur = first;
while ( cur->link != NULL)
{
prev=cur;
cur = cur->link;
}
printf (" item deleted at rear-end is %d", cur->info);
free(cur);
```

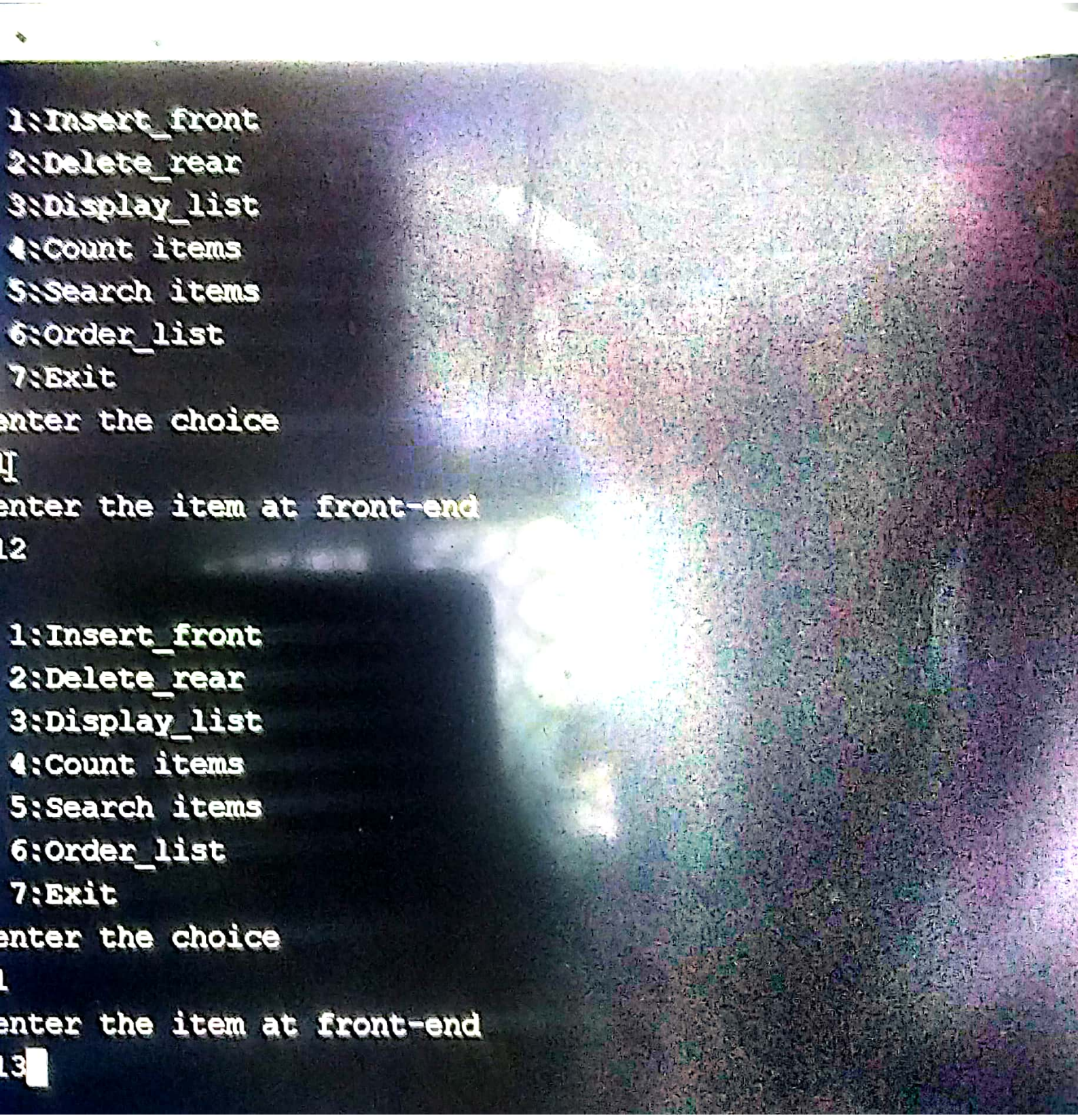
```
previous = NULL;
return first;
}
void display (NODE first)
{
NODE temp;
if (first == NULL)
printf ("list empty can't display\n");
for (temp = first; temp != NULL; temp = temp->link)
{
printf ("%d\n", temp->info);
}
}
int length (NODE first)
{
NODE cur;
int count = 0;
if (first == NULL)
return 0;
cur = first;
while (cur != NULL)
{
count ++;
cur = cur->link;
}
return count;
}
void search (int key, NODE first)
{
```

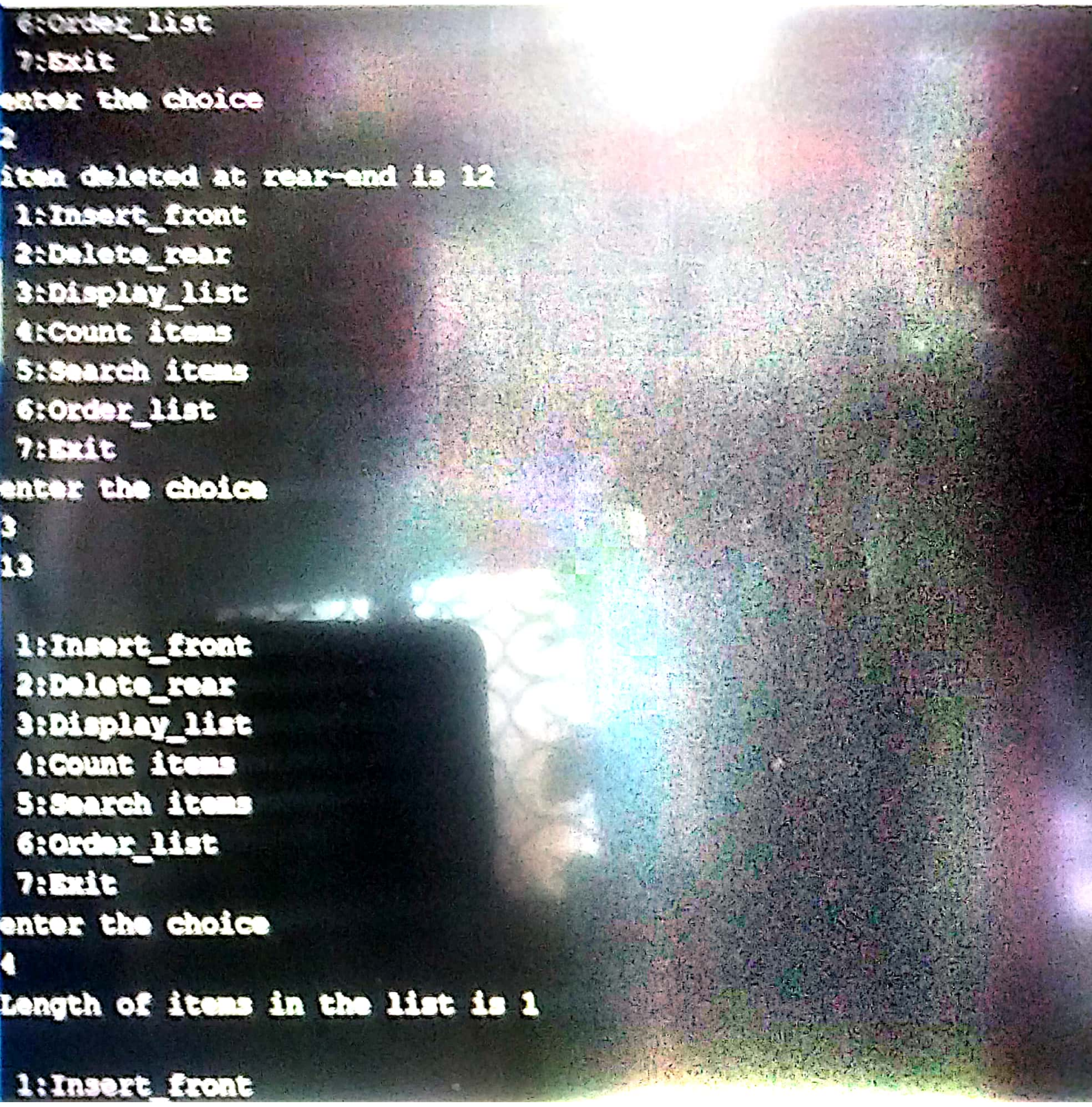
```
NODE cur;
if(first == NULL)
{
printf("list empty\n");
return;
}
cur = first;
while(cur != NULL)
{
if(key == cur->info)
break;
cur = cur->link;
}
if(cur == NULL)
{
printf("search unsuccessfull\n");
return;
}
printf("search successfull\n");
}
NODE asc(NODE first)
{
NODE prev=first;
NODE cur = NULL;
int temp;
if(first == NULL {
return 0; }
else {
while(prev != NULL) {
```

```
cur = prev->link;
while (cur != NULL) {
if(prev->info > cur->info) {
temp = prev->info
prev->info = cur->info
cur->info = temp;
}
cur = cur->link;
}
prev = prev->link;
   }
}
return first;
}
NODE del (NODE first).
{
NODE prev = first;
NODE cur = NULL;
 int temp;
 if (first == NULL)
{
return 0;
}
else {
while (prev != NULL){
cur = prev->link;
while (cur!= NULL) {
if (prev->info< cur->info) {
temp = prev->info;
```

```
prev->info = cur->info;
cur->info = temp;
}
cur = cur->link;
prev = prev->link;
}return first;
}
int main() {
int item, choice, count, key, option;
NODE first = NULL;
for(;;) {
printf("\n1: Insert_front\n2: Delete_rear\n3:
-list\n4:count-items\n5: search items\n6:
list\n7: exit\n");
printf("enter the choice\n");
scanf("%d", &choice);
switch(choice) {
case 1: printf("enter the item at front_end
        scanf("%d", &item);
        first = insert_front(first, item);
        break;
case 2: first = delete_rear(first); break;
case 3: display(first); break;
case 4: count = length(first);
        printf("length = %d\n", count); br
case 5: printf("enter the item to be searched
        scanf("%d", &key);
        search(key, first);
        break;
```

```
case 6: printf("\n1: ascending ordered list\n
2: descending Ordered_list\n");
 scanf("%d", &option);
 if (option==1)
{
first = asc(first);
display(first);
}
else{
first= des(first);
  display(first);
  } break;
default : exit(0);
  }
  return 0;
}
```





```
enter the item to be searched
23
Search is unsuccessful

 1:Insert_front
 2:Delete_rear
 3:Display_list
 4:Count items
 5:Search items
 6:Order_list
 7:Exit
enter the choice
6

1:ascending ordered_list
2:descending ordered_list
1
13

 1:Insert_front
 2:Delete_rear
 3:Display_list
 4:Count items
 5:Search items
 6:Order_list
 7:Exit
enter the choice
```