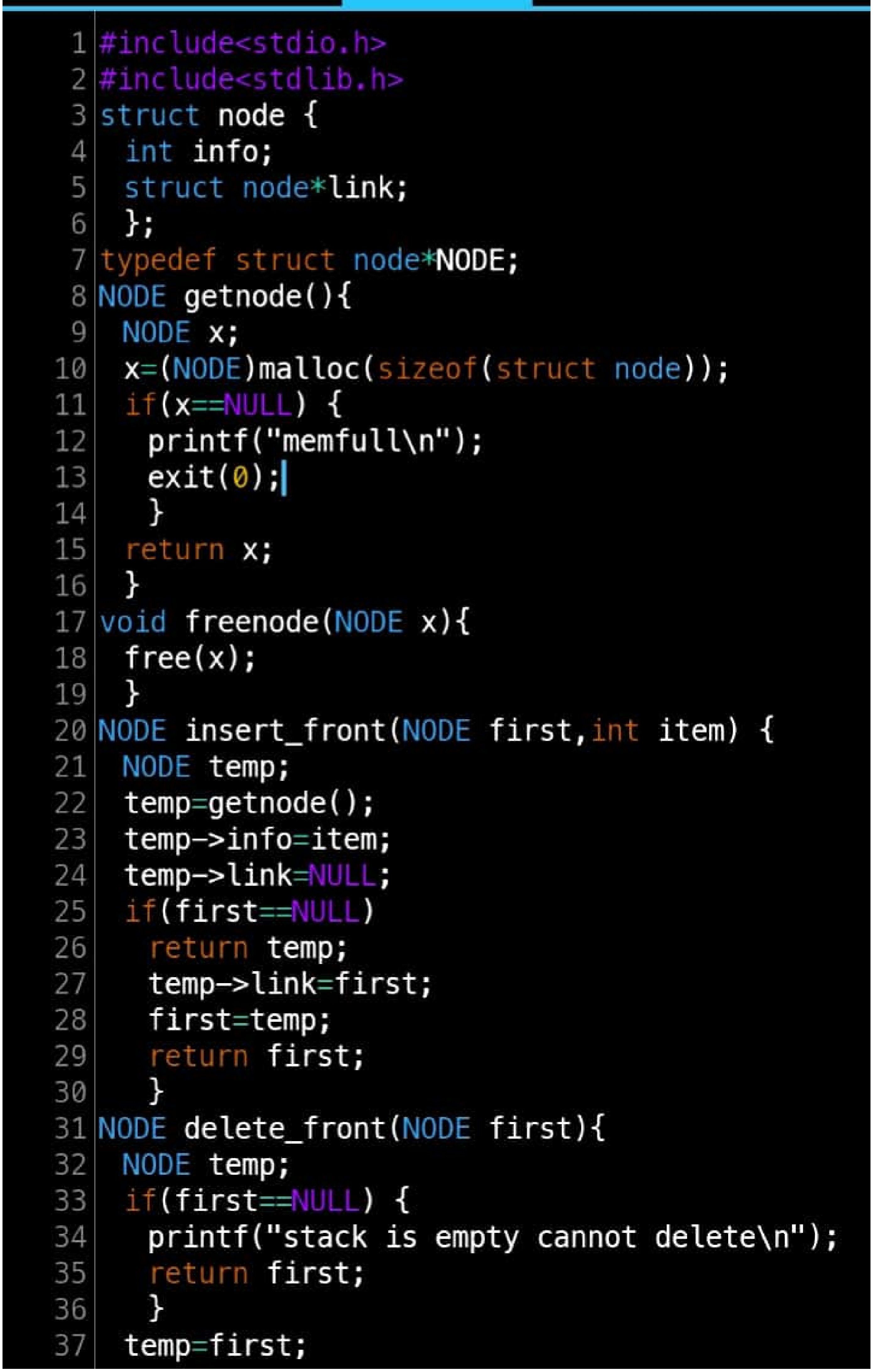
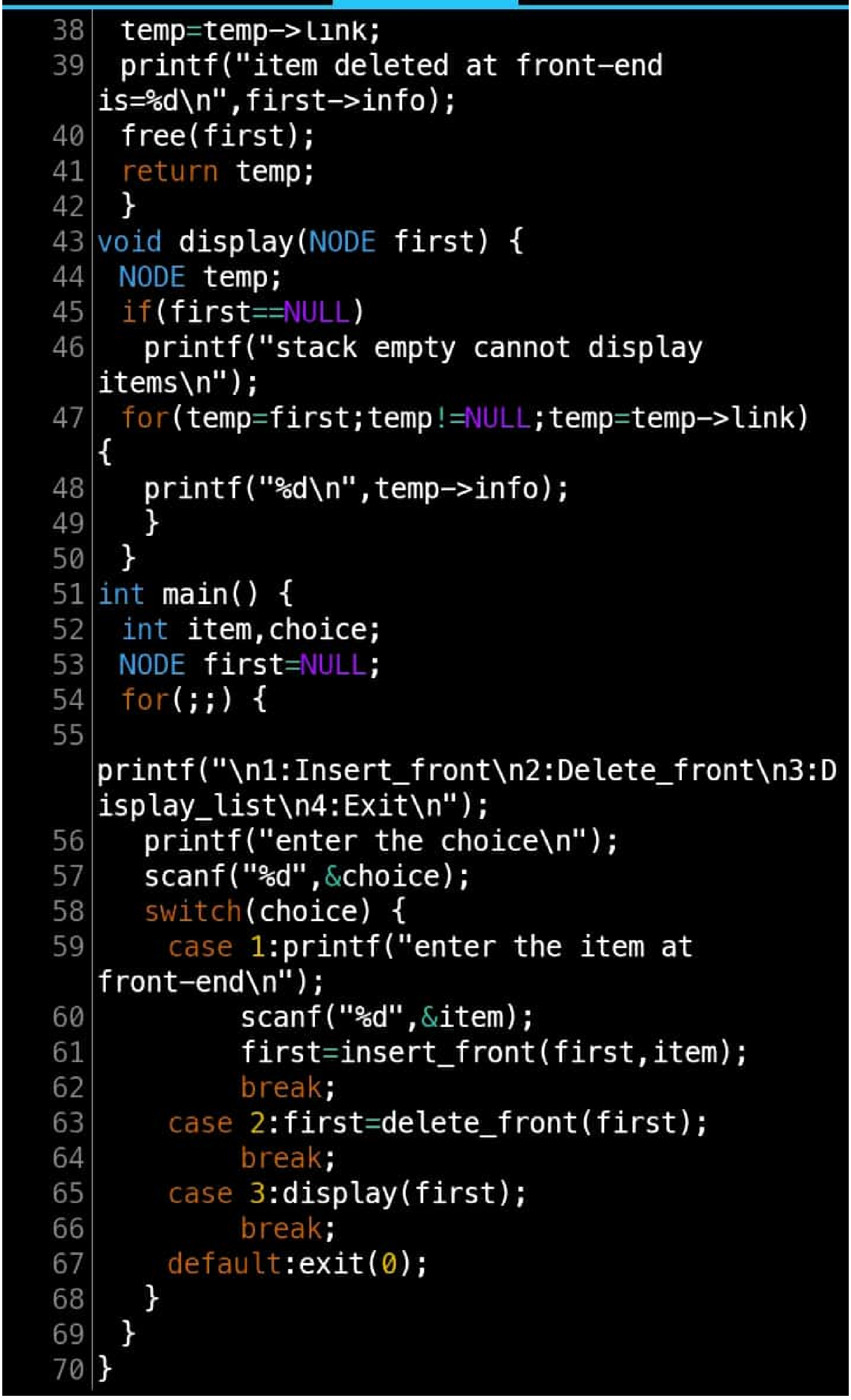
```
# include <stdio.h>
 struct node
 {
  int info;
  struct node *link;
 }
typedef struct node *NODE;
     NODE getnode()
     {
      NODE x;
      x=(NODE) ~~sizeof~~ malloc (sizeof(NODE));
      if (x==NULL)
      {
       printf("Memory full \n");
       exit(0);
      }
      return x;
     }
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

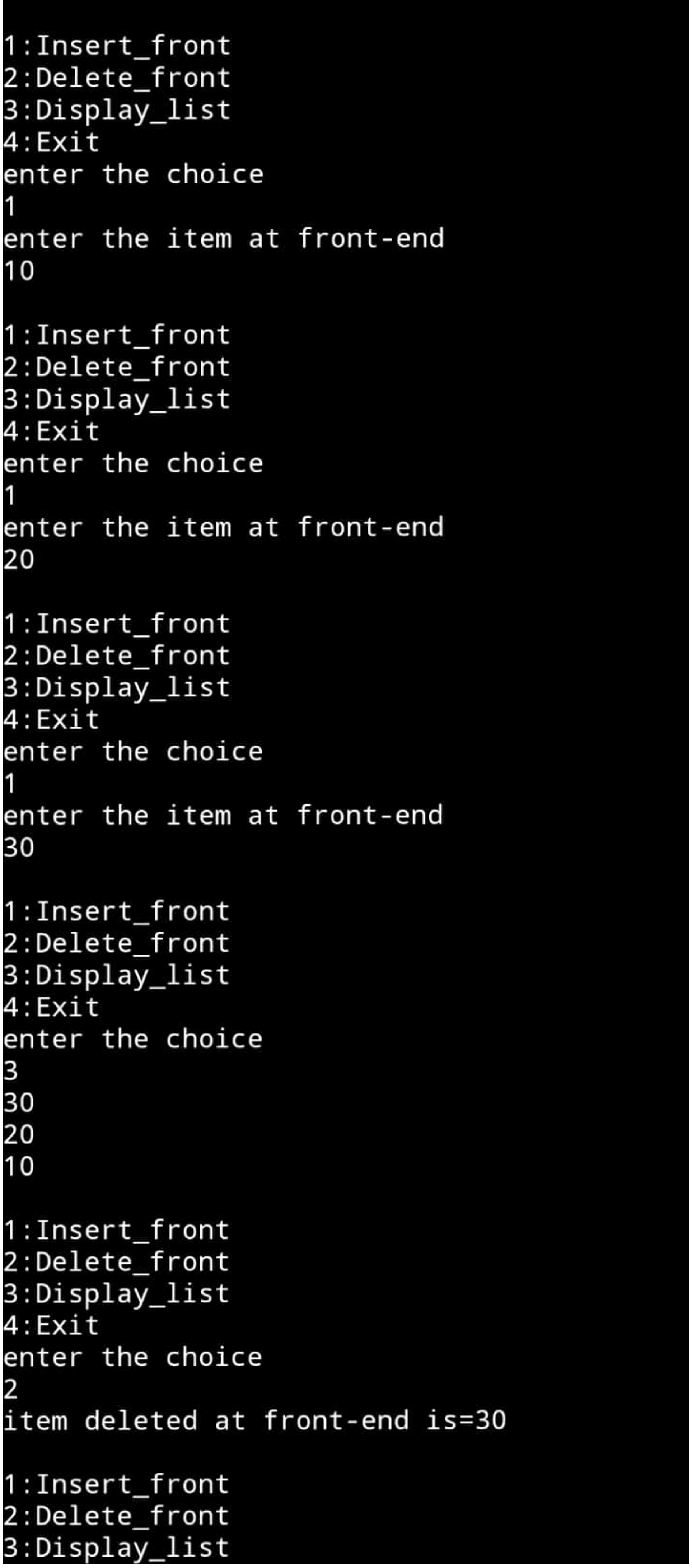
```
     first = temp ;
      return first ;
  }
NODE pop (NODE first)
{
  NODE temp ;
  if (first = = NULL)
  {
   printf ("Stack under flow \n");
   ~~exit(0)~~; return first ;
  }
  temp = first ;
  temp = temp -> link ;
  printf ("item deleted ~~@~~at front end is = %d \n", first->info);
  ~~ret~~ free (first);
       return temp ;
  }
void display ()
{
  NODE temp ;
  if (first = = NULL)
    printf ( "Stack empty \n");
   for (temp = first ; temp != NULL ; temp = temp -> link)
    ~~@~~ printf ("%d \n", temp->info);
    }
int main ()
{
  int item , choice ;
  NODE first = NULL ;
  for(;;)
   {
  printf ("\n1. Push \n2. Pop \n 3. display \n4. Exit \n);
printf ("Enter your choice \n);
scanf ("%d \n", &choice);
switch (choice)
{
case 1: printf ("Enter the item \n");
        scanf ("%d \n", &item);
        first = push (first, item);
        break;
case 2: first = pop (first);
        break;
case 3: display ();
        break;
~~case 4~~ : default : exit (0);
             break;
      }
     }
    }
```



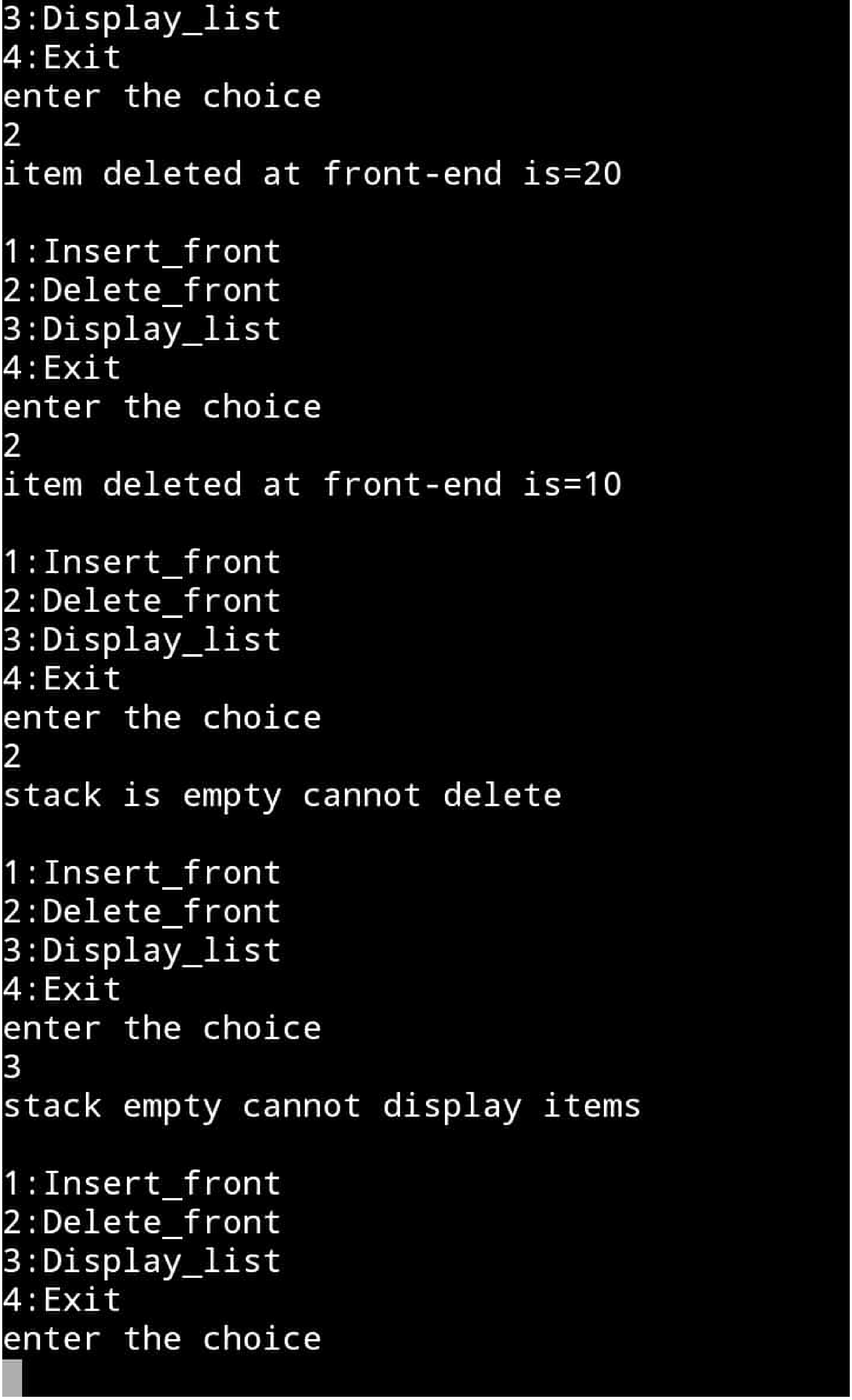
```c
#include<stdio.h>
#include<stdlib.h>
struct node {
 int info;
 struct node*link;
 };
typedef struct node*NODE;
NODE getnode(){
 NODE x;
 x=(NODE)malloc(sizeof(struct node));
 if(x==NULL) {
   printf("memfull\n");
   exit(0);
   }
 return x;
 }
void freenode(NODE x){
 free(x);
 }
NODE insert_front(NODE 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_front(NODE first){
 NODE temp;
 if(first==NULL) {
   printf("stack is empty cannot delete\n");
   return first;
   }
 temp=first;
```



```c
 temp=temp->link;
 printf("item deleted at front-end is=%d\n",first->info);
 free(first);
 return temp;
 }
void display(NODE first) {
 NODE temp;
 if(first==NULL)
   printf("stack empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link) {
   printf("%d\n",temp->info);
   }
 }
int main() {
 int item,choice;
 NODE first=NULL;
 for(;;) {
printf("\n1:Insert_front\n2:Delete_front\n3:Display_list\n4:Exit\n");
   printf("enter the choice\n");
   scanf("%d",&choice);
   switch(choice) {
     case 1:printf("enter the item at front-end\n");
            scanf("%d",&item);
            first=insert_front(first,item);
            break;
     case 2:first=delete_front(first);
            break;
     case 3:display(first);
            break;
     default:exit(0);
   }
 }
}
```



```
1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
10

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
20

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
30

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
3
30
20
10

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=30

1:Insert_front
2:Delete_front
3:Display_list
```



```
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=20

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=10

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
2
stack is empty cannot delete

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
3
stack empty cannot display items

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
```