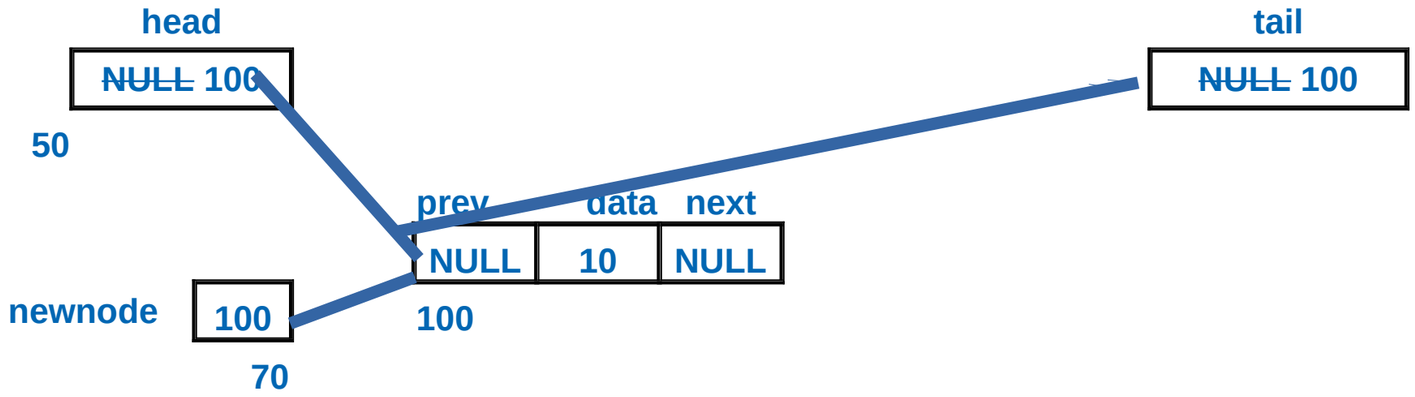
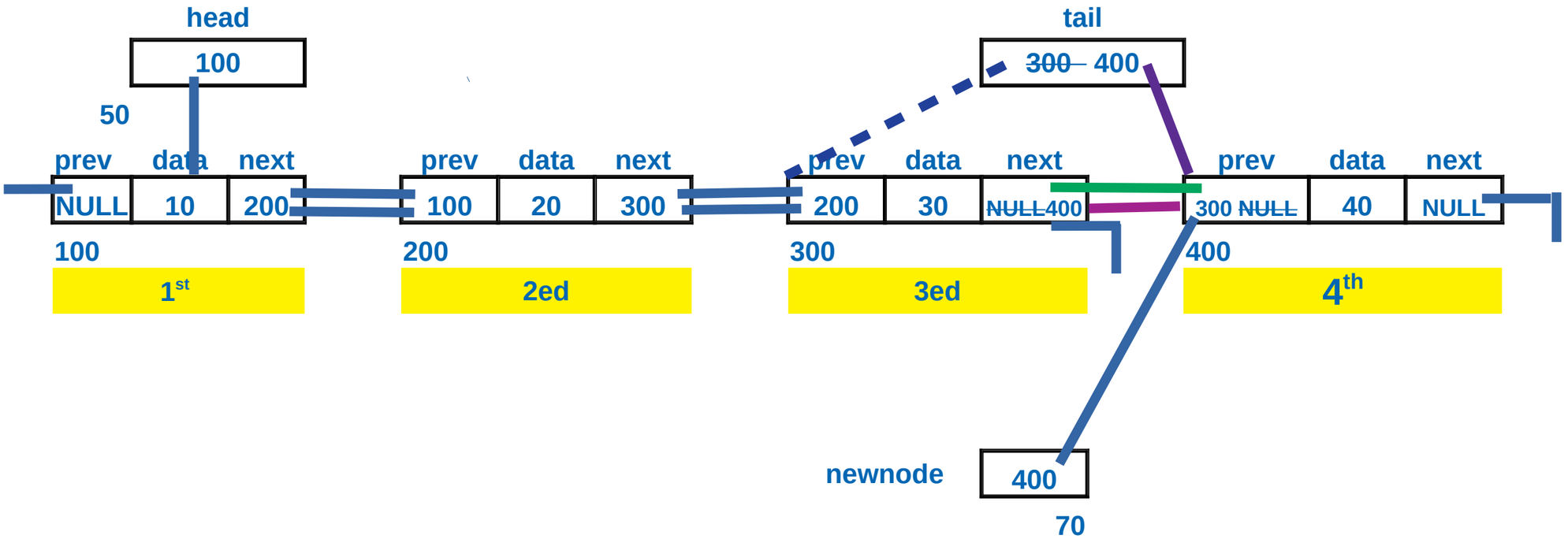
```
void add_last(int value)
{
   node_t * newnode= create_node
   if(head==NULL)
   {
      head=newnode;
      tail= newnode;
   }
   else
   {
      newnode->prev=tail;
      tail->next=newnode;
      tail= newnode;
   }
}
```

Case 1 : add last when list is empty



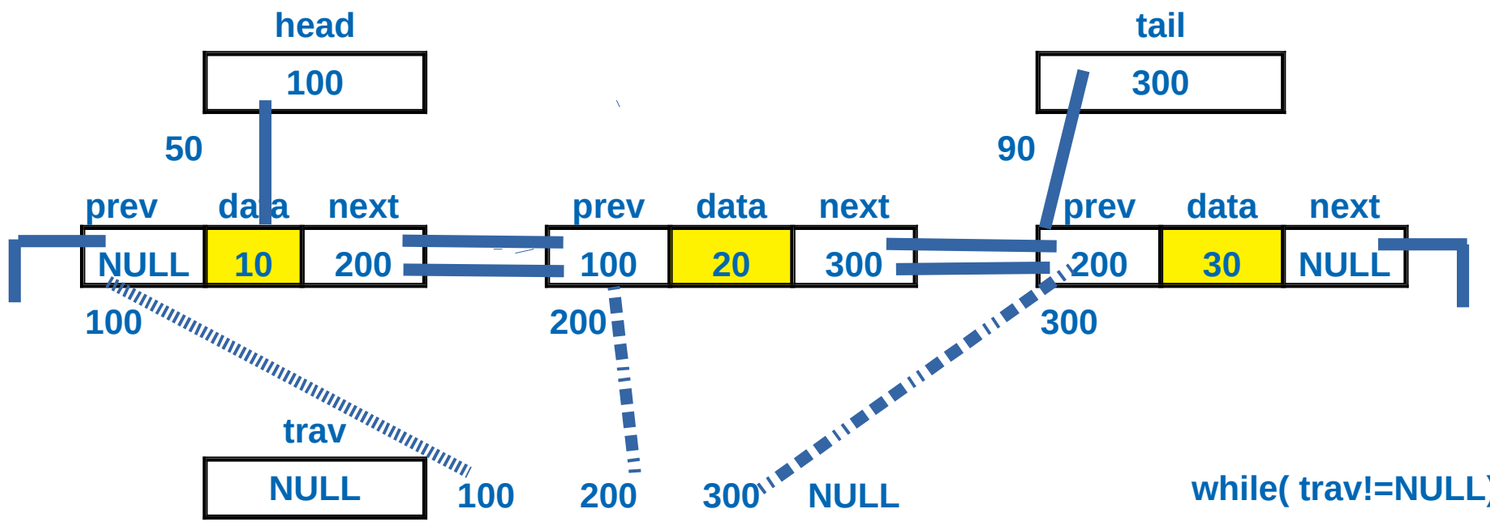
Case 2 : add last when list is has multiple nodes



```
void display_list()
{
   if(head== NULL)
      printf("\n List is empty");
   else
   {
      printf("\n Forword display\n");
      node_t *trav=head;  // address of 1st
      while (trav!=NULL)
      {
         printf("%d----->", trav->data);
         trav= trav->next;
      }

      printf("\n backword display\n");
      trav=tail; // address of last node
      while (trav!=NULL)
      {
         printf("<----%d", trav->data);
         trav= trav->prev;
      }

   }


}
```

Case 1 : display when list is empty

Case 2 : display when list has multiple nodes



forword display

10 20 30

backword display

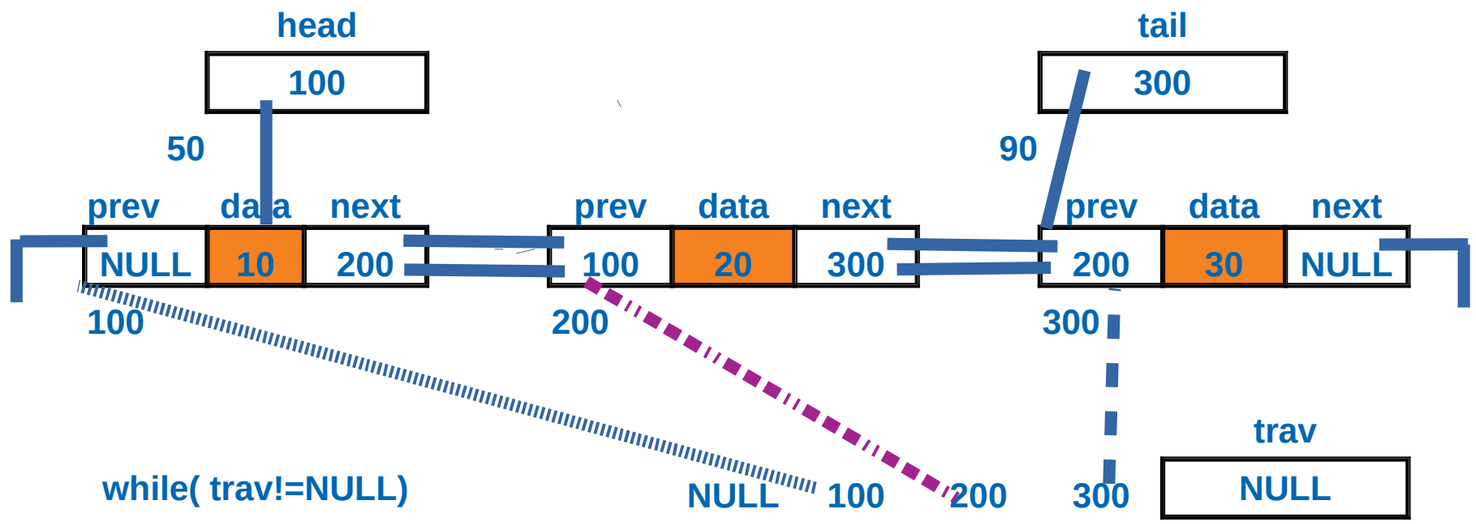
while(trav!=NULL)

while(NULL!=NULL) **FALSE**

```
void display_list()
{
   if(head== NULL)
      printf("\n List is empty");
   else
   {
      printf("\n Forword display\n");
      node_t *trav=head;  // address of 1st
      while (trav!=NULL)
      {
         printf("%d----->", trav->data);
         trav= trav->next;
      }

      printf("\n backword display\n");
      trav=tail; // address of last node
      while (trav!=NULL)
      {
         printf("<----%d", trav->data);
         trav= trav->prev;
      }

   }


}
```

Case 1 : display when list is empty

Case 2 : display when list has multiple nodes



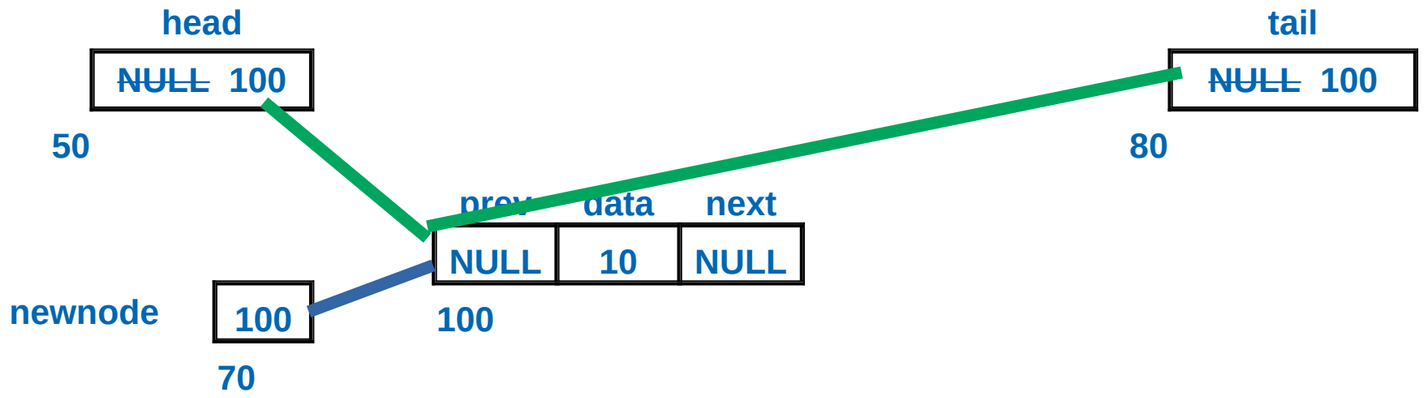
forword display

backword display

30 20 10

```
void add_first(int value)
{
   node_t* newnode= create_node(
   if(head==NULL)
   {
      head=newnode;
      tail=newnode;
   }
   else
   {
      newnode->next=head;
      head->prev=newnode;
      head=newnode;
   }
}
```

Case 1 : add first when list is empty



Case 2 : add frist when list is has multiple nodes