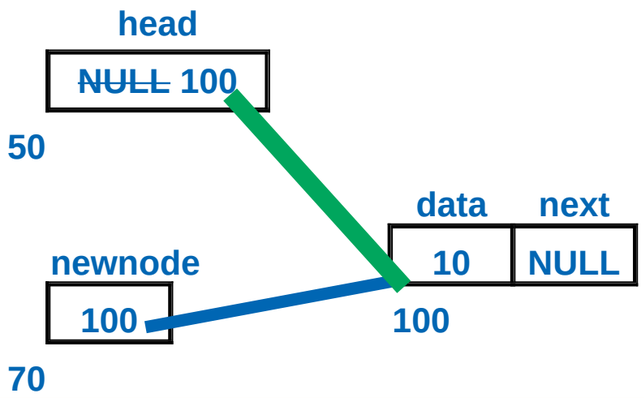
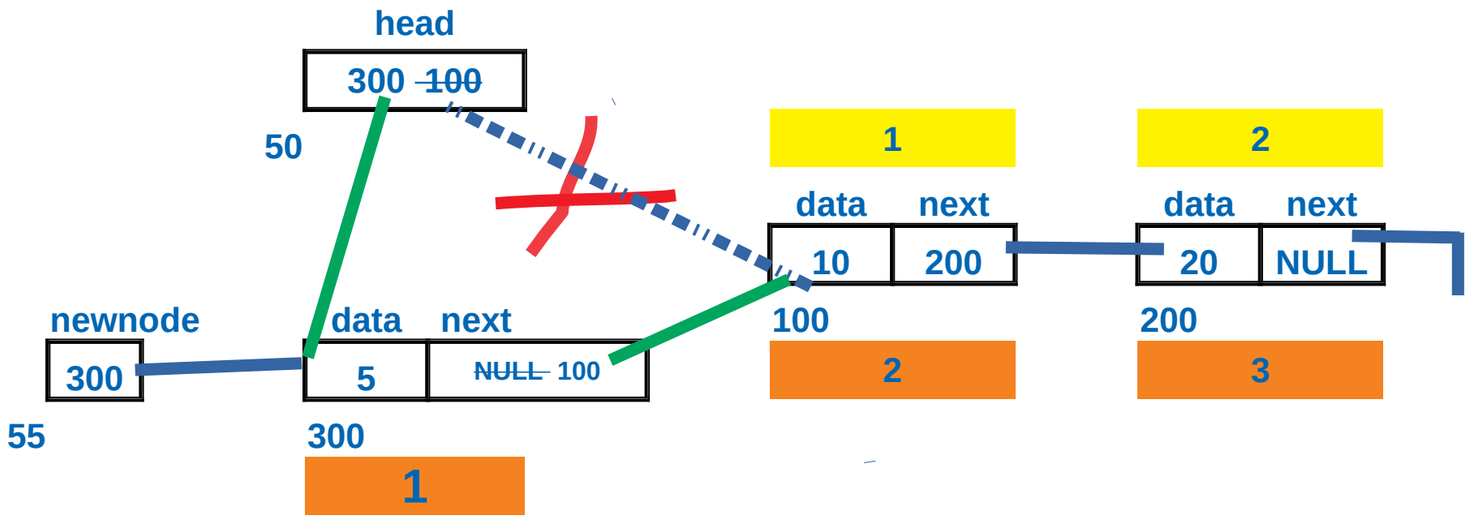
```
void AddFirst(int value)
{
   node_t *newnode=NULL;
   newnode= CreateNode(value);
   if(head==NULL)
   head=newnode;
   else
   {
      newnode->next=head;
      head=newnode;
   }
   return ;
}
```

Case 1 : add first when linked list is empty



Addfrist(10);

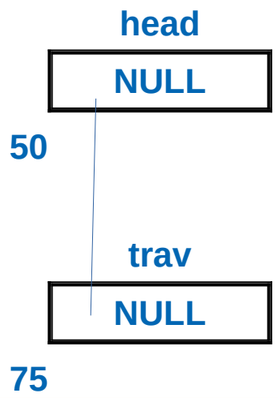
Case 2 : add first when list is has multiple nodes



```
void DisplayList()
{
   node_t * trav=NULL;
   trav=head;
   if(trav==NULL)
   {
   printf("\n list is empty");
   }
   else
   {
      while(trav!=NULL)
      {
      printf("%d--->", trav->data);
      trav=trav->next;
      }
   }
   printf("\n");
   return;
}
```

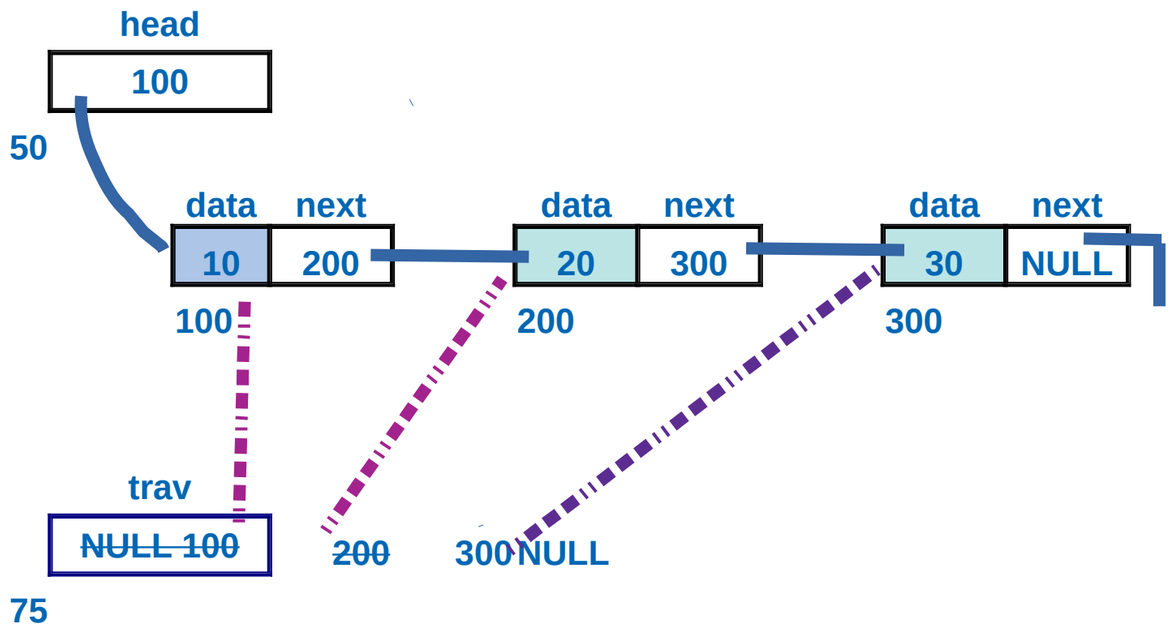
10 20 30

Case 1 : display when list is empty



list is empty

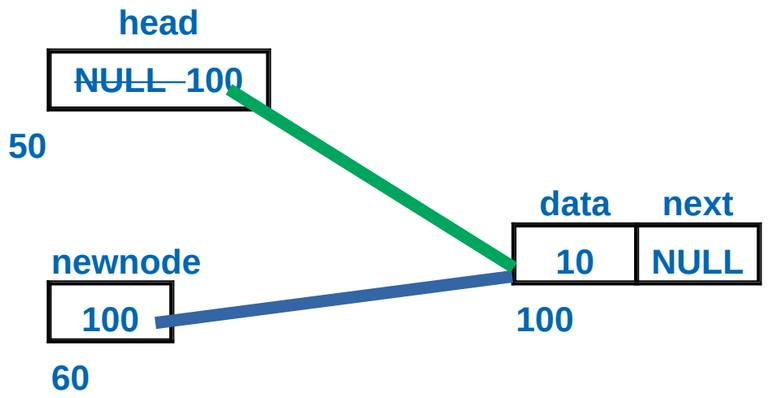
Case 2 : display when list has multiple nodes



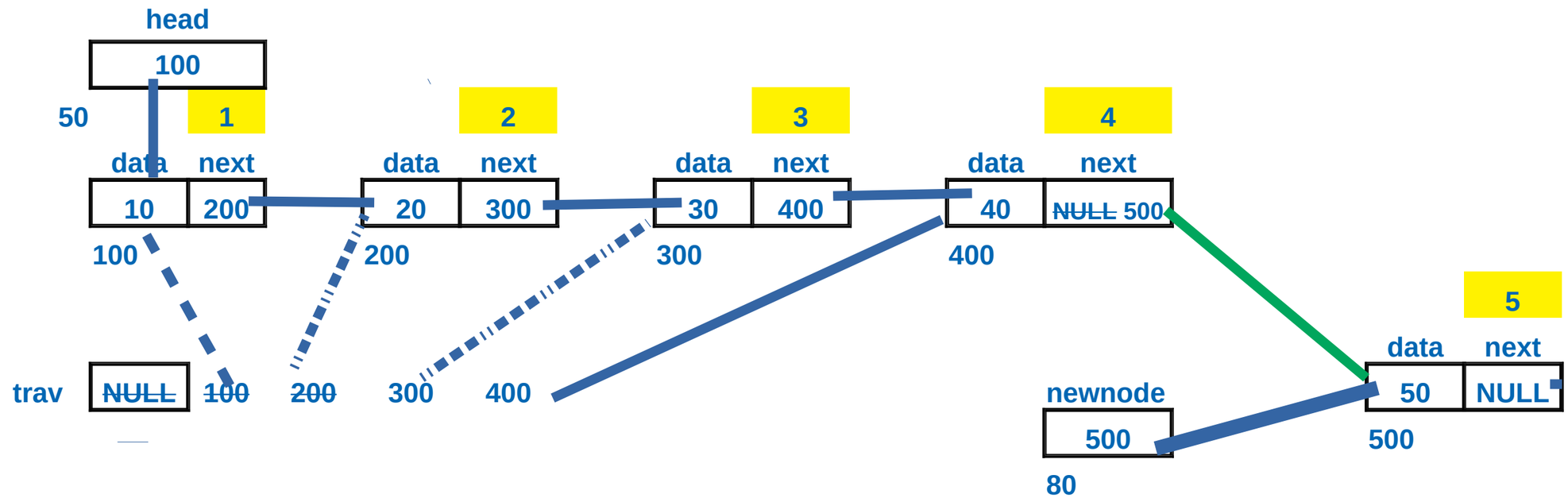
~~while(100!=NULL)~~	TRUE
While(200!=NULL)	TRUE
While(300!=NULL)	TRUE
While(NULL!=NULL)	FALSE

```
void AddLast(int value)
{
   node_t *newnode=NULL;
   newnode=CreateNode(value);
   if(head==NULL)
      head=newnode;
   else
   {
      NODE *trav=head;
      while(trav->next!=NULL)
      {
         trav=trav->next;
      }
      trav->next=newnode;
   }
   return ;
}
```

Case 1 : add last when list is empty



Case 2 : add last when list is has multiple nodes



While(200!=NULL) TRUE

While(300!=NULL) TRUE

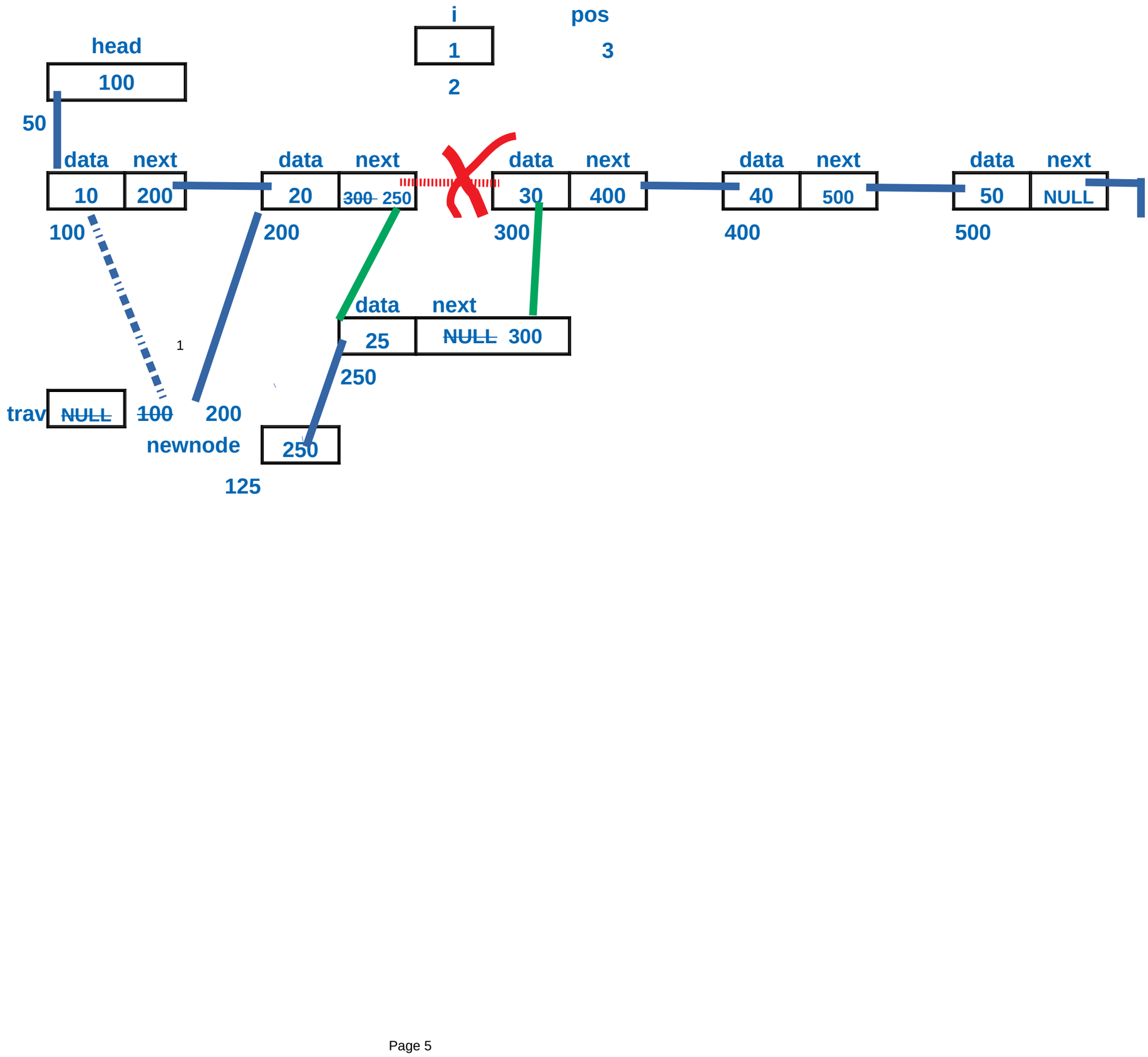
While(400!=NULL) TRUE

While(NULL!=NULL) FALSE

AddLast

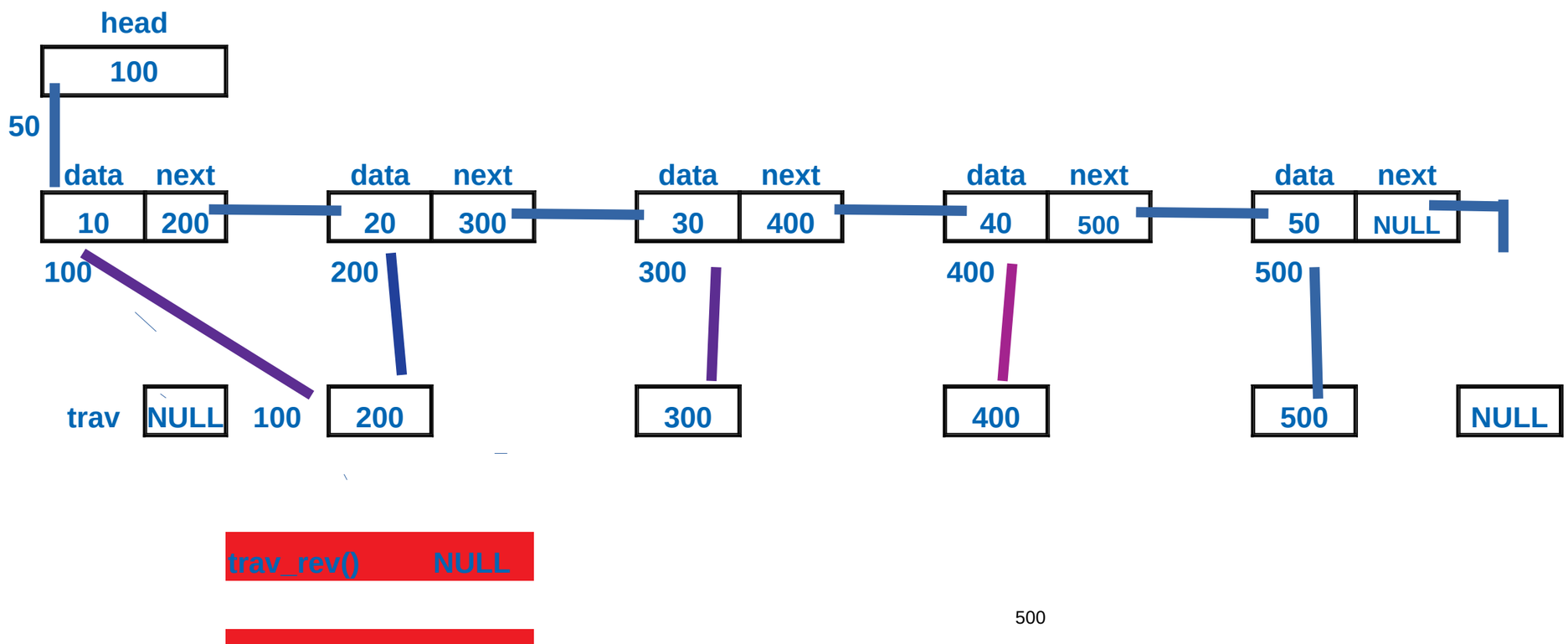
```
void AddAtPosition(int position, int valu
{
node_t *newnode=NULL;
node_t *trav=NULL;
int i;
if(position==1)
AddFirst(value);
else if( (CountNodes()+1)==position)
AddLast(value);
else
{
newnode= CreateNode(value);
trav=head;
for(i=1; i<position-1; i++)
{
trav=trav->next;
}
newnode->next=trav->next;
trav->next=newnode;
}
return;
}
```

Case 1 : add at 3 rd position when list is has 5 nodes



Case 1 : Print Linked List In Reverse

```
void trav_rev(node_t * trav)
{
   if(trav== NULL)
      return;
   else
      trav_rev(trav->next);
   printf("%5d-->", trav->data);
}
```



50 40 30 20 10