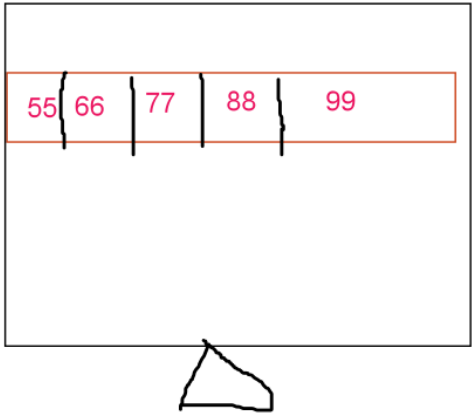
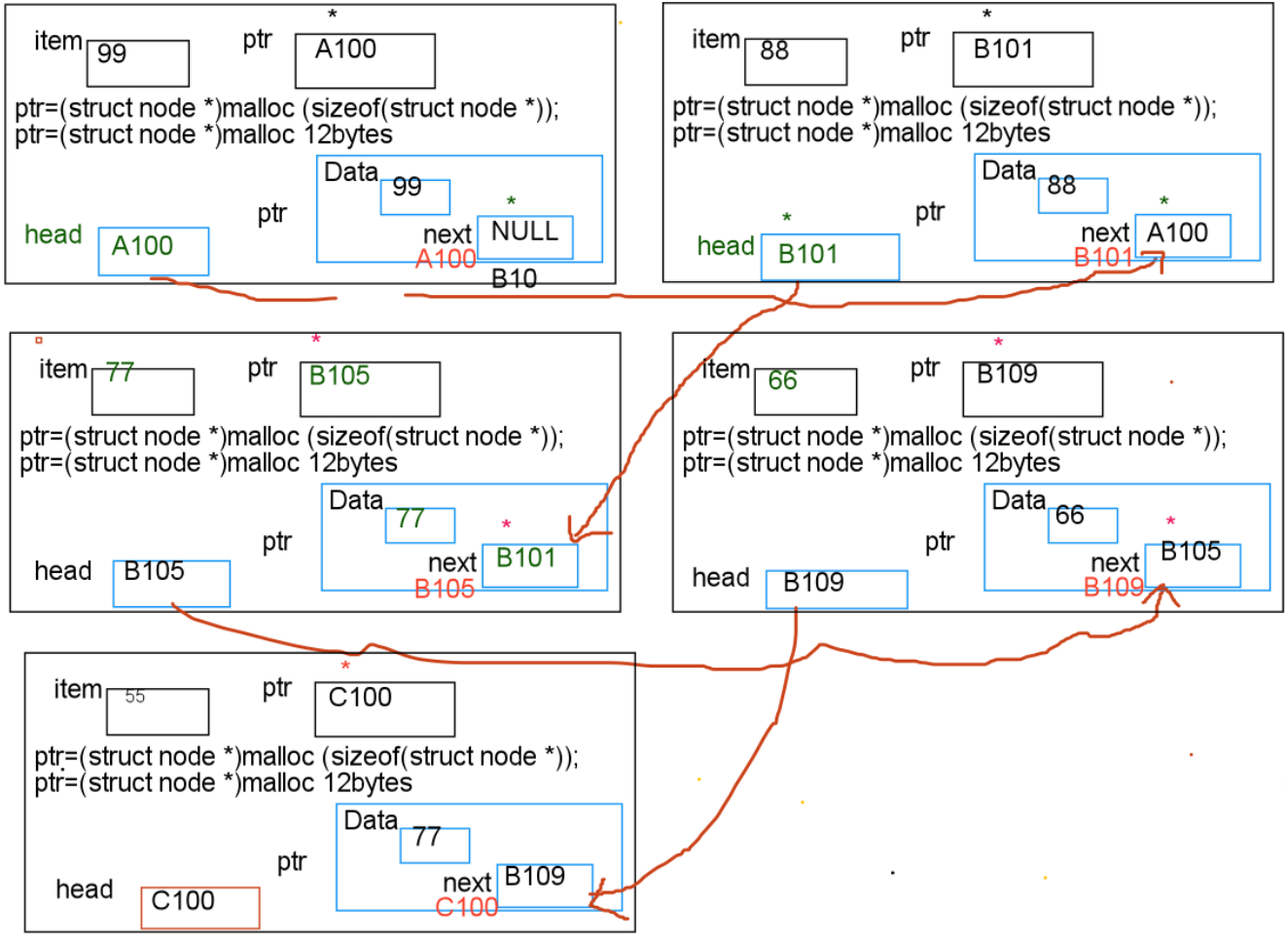
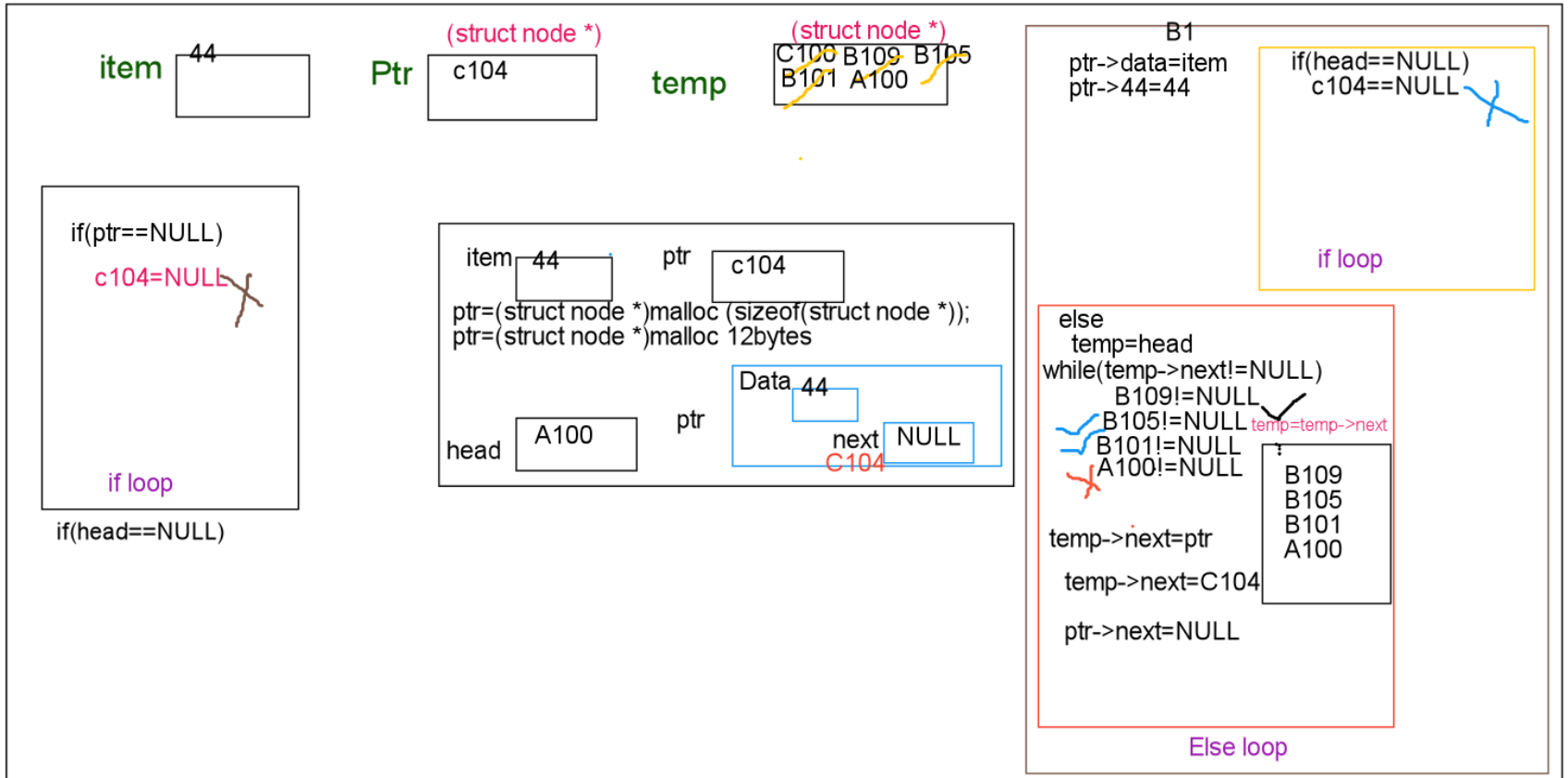
```
void beginsert()
  {
     struct node *ptr;
     int item;
     ptr = (struct node *) malloc(sizeof(struct node *));
     if(ptr == NULL)
     {
        printf("\nOVERFLOW");
     }
     else
     {
        printf("\n Enter value\n");
        scanf("%d",&item);
        ptr->data = item;
        ptr->next = head;
        head = ptr;
        printf("\nNode inserted");
     }

  }
```







Void Last insert()

```
void randominsert()
  {
    int i,loc,item;
    struct node *ptr, *temp;
    ptr = (struct node *) malloc (sizeof(struct node));
    if(ptr == NULL)
    {
      printf("\nOVERFLOW");
    }
    else
    {
      printf("\nEnter element value");
      scanf("%d",&item);
      ptr->data = item;
      printf("\nEnter the location after which you want to insert ");
      scanf("\n%d",&loc);
      temp=head;
      for(i=0;i<loc;i++)
      {
        temp = temp->next;
        if(temp == NULL)
        {
          printf("\ncan't insert\n");
          return;
        }

      }
      ptr ->next = temp ->next;
      temp ->next = ptr;
      printf("\nNode inserted");
    }
  }
```

i 0 1 2

temp B109 B105

item 22

loc 2

ptr= (struct node *)malloc (sizeof(struct node))
ptr=(struct node *)malloc 12 bytes

ptr → data 22 | next B101 D100

if(ptr==NULL)
c104==NULL)

ifloop

for(i=0;i<loc;i++)

i<loc i++

0<2

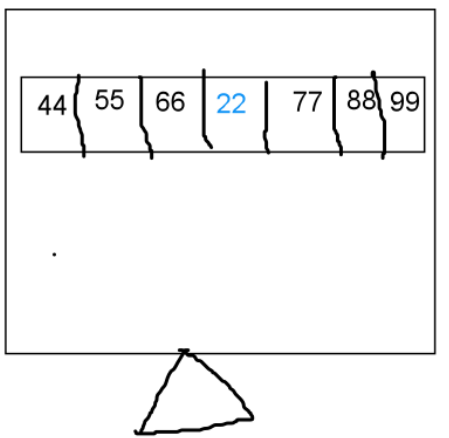
1<2

2<2

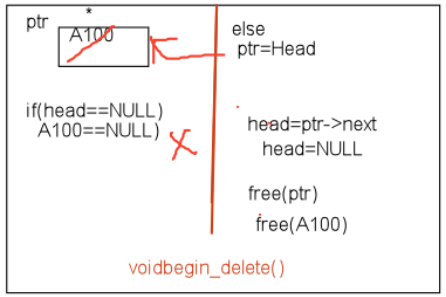
temp=temp->next
B109
B105

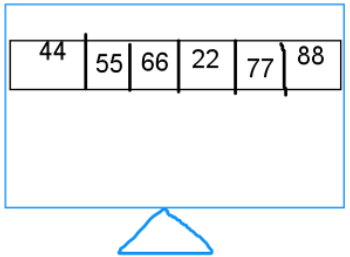
if(temp==NULL)
B109==NULL
B105==NULL

else loop

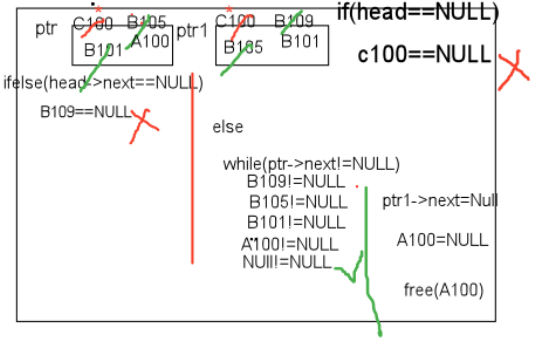


```
void begin_delete()
   {
      struct node *ptr;
      if(head == NULL)
      {
         printf("\nList is empty\n");
      }
      else
      {
         ptr = head;
         head = ptr->next;
         free(ptr);
         printf("\nNode deleted from the begining ...\n");
      }
   }
```

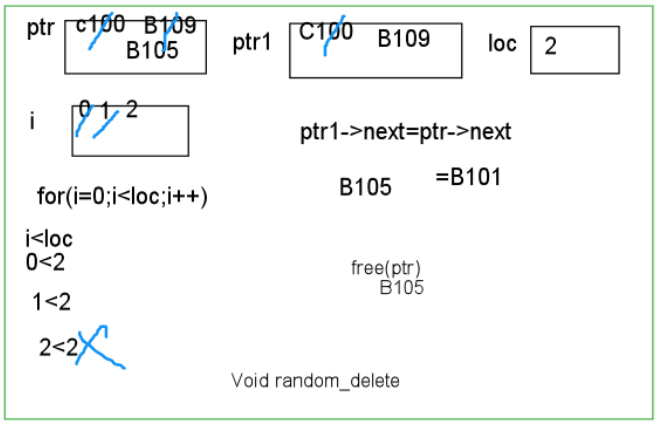


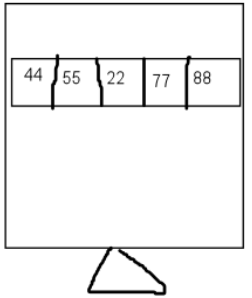



```
void last_delete()
   {
      struct node *ptr,*ptr1;
      if(head == NULL)
      {
         printf("\nlist is empty");
      }
      else if(head -> next == NULL)
      {
         head = NULL;
         free(head);
         printf("\nOnly node of the list deleted ...\n");
      }

      else
      {
         ptr = head;
         while(ptr->next != NULL)
         {
            ptr1 = ptr;
            ptr = ptr ->next;
         }
         ptr1->next = NULL;
         free(ptr);
         printf("\nDeleted Node from the last ...\n");
      }
   }
```

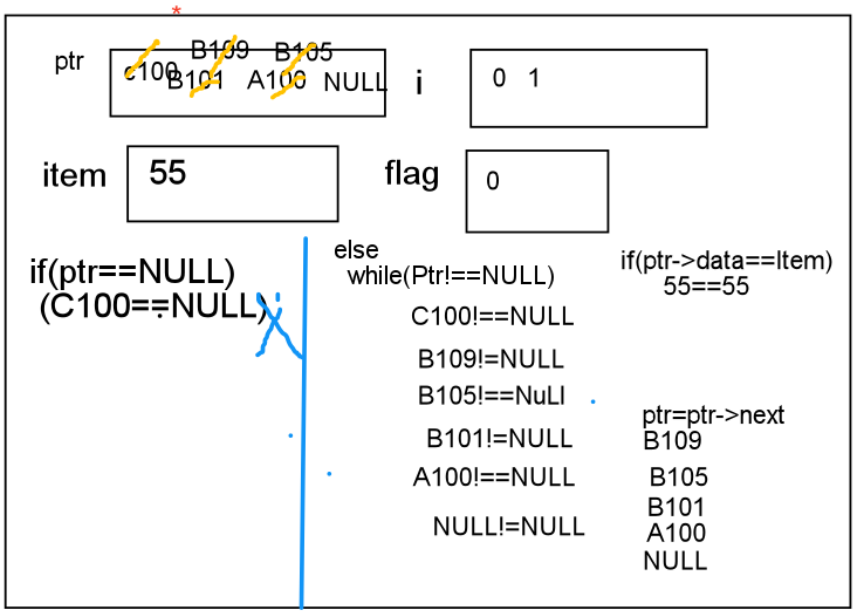



```
void random_delete()
   {
      struct node *ptr,*ptr1;
      int loc,i;
      printf("\n Enter the location of the node after which you want to perform deletion \n");
      scanf("%d",&loc);
      ptr=head;
      for(i=0;i<loc;i++)
      {
         ptr1 = ptr;
         ptr = ptr->next;

         if(ptr == NULL)
         {
            printf("\nCan't delete");
            return;
         }
      }
      ptr1 ->next = ptr ->next;
      free(ptr);
      printf("\nDeleted node %d ",loc+1);
   }
```

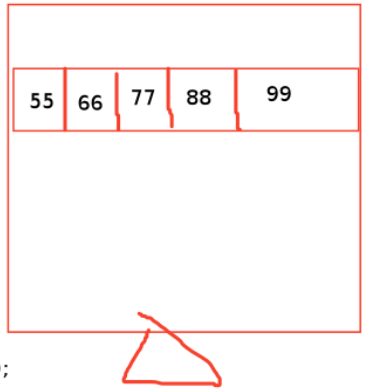


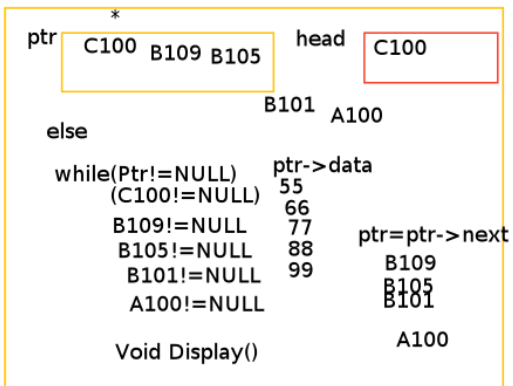


```
void search()
  {
     struct node *ptr;
     int item,i=0,flag;
     ptr = head;
     if(ptr == NULL)
     {
        printf("\nEmpty List\n");
     }
     else
     {
        printf("\nEnter item which you want to search?\n");
        scanf("%d",&item);
        while (ptr!=NULL)
        {
           if(ptr->data == item)
           {
              printf("item found at location %d ",i+1);
              flag=0;
           }
           else
           {
              flag=1;
           }
           i++;
           ptr = ptr -> next;
        }
     }

        if (flag==1)
        {
           printf("Item not found\n");
        }


}
```



```
void display()
  {
    struct node *ptr;
    ptr = head;
    if(ptr == NULL)
    {
       printf("Nothing to print");
    }
    else
    {
       printf("\n printing values\n ");
        printf("\n--------------------\n");

       while (ptr!=NULL)
       {
          //printf("------");
          printf("| %d ",ptr->data);
          //printf("-----");
          ptr = ptr -> next;


       }
        printf("|\n----------------------\n");
    }
  }
```






```
choice 0 1 1 1 1 1 2 3 4 5 6 7 8

While(Choice!=9)
      (0!=9)
switch(choice)
      {
        case 1:
        beginsert();
        break;

        case 2:
        lastinsert();
        break;

        case 3:
        randominsert();
        break;

        case 4:
        begin_delete();
        break;

        case 5:
        last_delete();
        break;

        case 6:
        random_delete();
        break;

        case 7:
          search();
        break;

        case 8:
        display();
        break;

        case 9:
        exit(0);
```

```
Enter your choice:1
Enter Value:99
 node inserted succesfully!

Enter your choice:1
Enter Value:88
node inserted successfully!

Enter your choice:1
Enter Value:77
node inserted successfully!

Enter your choice:1
Enter Value:66
node inserted successfully!

Enter your choice:1
Enter Value:55
node inserted successfully!

Enter Your choice:2
 Enter Value:44
node inserted succesfully!

Enter your choice:3
 Enter value:22
 Enter the loction after which you want insert2

Enter your choice:4
node deleted beging...

Enter your choice:5
 node deleted last...
Enter your choice:6
Enter the location of the node after which you want to perform deletion :2
deleted node is3

 Enter your choice:7
enter the element you want serch:77
element found

   Enter your choice:8
```