Page No:
Date: / /

Name : S Skanda                                        Date : 23/11/2020
USN : 1BM19CS137

```
→ #include<stdio.h>
  #include<conio.h>
  #include <stdlib.h>
  #include <process.h>
→ struct node
  { int info
    struct node *link;
  ~~typedef NODE~~
  typedef struct node *NODE;

→ NODE getnode ()
  { NODE x;
      x= (NODE)malloc (sizeof(struct node));
      if (x= =NULL )
      { printf("memory 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 front (NODE first)
{ NODE temp;
  if (first == NULL)
  { printf ("List is empty, cannot delete");
    return first;
  }

  temp = first;
  temp = temp -> link;
  printf ("item deleted at front is =%d", first->info);
  free (first);
  return temp;
}
NODE insert rear (NODE first, int item)
{ NODE temp, cur;
  temp = getnode ();
  temp -> info = item;
  temp -> link = NULL
  if (first == NULL) return temp;
  cur = first;
  while (cur -> link != NULL)
       cur = cur -> link;
  cur -> link = temp;
  return first;
}


NODE delete_rear (NODE first)
{ NODE cur, prev;
  if (first == NULL)
  { printf ("List is empty can't delete");
    return first;
  }
```

```
if (first -> lint == 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 is %d \n", cur->info);
free (cur);
prev -> link = NULL;
return first;
}
NODE delete-info (int key, NODE first)
{ NODE prev, cur;
  if (first == NULL).
  { printf ("list is empty \n"); return NULL; }
  if ( key == first -> info)
  { cur = first;
    first = first -> link;
    free(cur);
    return first;
  }

  prev = NULL
  cur = first
  while (cur != NULL)
  { if (key = cur->info) break;
    prev = cur;
    cur = cur -> ink
  }.
```

```
   if (cur == NULL)
   { printf("search unsuccessfull\n");
     return first;
   }

   prev->link = cur->link;
   printf("Key deleted is %d", cur->info);
   free(cur);
   return first;
}  }


NODE insert_pos (int item, int pos, NODE first)
{ NODE temp, prev, cur;
   int count;
   temp=getnode();
   temp->info= item;
   temp->link= NULL;
   if(first == NULL && Pos == 1)
   {. return temp;
   if (first == NULL)
    { printf("invalid position\n");
      return first;
    }
   if(pos ==1 )
    { temp->link= first;
      return temp;
    }
   count = 1;
   prev = NULL
   cur = first;
```

```
while (cur != NULL && count != pos)
{ prev = cur;
  cur = cur->link;
  count++;
}
if (count == pos)
 { prev->link = temp;
   temp->link = cur;
   return first;
 }
void display(NODE first)
{ NODE temp;
  if(first == NULL)
  printf("List is empty\n");
  else
  { printf("Contents of list:\n");
    for (temp = first; temp != NULL; temp = temp->link)
    {   printf("%d  ", temp->info);
    }
  }
}
int main()
{int item, choice, key, pos;
 NODE first = NULL;
 for (;;)
 printf("\n 1: insert_front
           \n 2: delete_front
           \n 3: insert rear
           \n 4: delete_rear
           \n 5: insert_pos
           \n 6; delete_specified
           \n 7: display
           \n 8; exit \n");
```