|  |  |  |
| --- | --- | --- |
| **Name: SANJANA.S** | **SRN: PES2UG20CS310** | **Section: H** |
| **Date: 17-06-2021** | **Week Number: 7** |

|  |  |
| --- | --- |
| **1** | Define a structure called cricket that will describe the following information:  player name  team name  batting average  Using cricket, declare an array player with 5 elements and write a program to read the information about all the 5 players and print a team-wise list containing names of player with their batting average.Write functions for the following:  i) Read the information of all the 5 players  ii)Sorting the players  iii)Displaying team-wise list containing names of player with their batting average  **Input:**  Enter data of 5 players  Enter PName TName BAvg for player-1 = sachin  India  98  Enter PName TName BAvg for player-2 = Rahul  India  45  Enter PName TName BAvg for player-3 = Jonty  Australia  89  Enter PName TName BAvg for player-4 = Imran  pakistan  75  Enter PName TName BAvg for player-5 = Shen  Australia  29  **Output:**  After teamwise sorting... Player list is  Jonty Australia 89.00  Shen Australia 29.00  sachin India 98.00  Rahul India 45.00  Imran pakistan 75.00 |
|  | **Program:**  **1Prog.c:**    #include<stdio.h>  #include<stdlib.h>  #include<string.h>  struct cricket{  char \*player\_name;  char \*team\_name;  int batting\_average; *//this shall be the day where i start taking meaningful varibles names.*  }players[5];  void read5(){  char temppname[50],temptname[50];  int tempbatavg;  for(int i=0;i<5;i++)  {  printf("Player %d : \n",i+1);  printf("Enter player name : ");  scanf(" %[^\n]%c",temppname);  (players+i)->player\_name=(char \*)malloc(50);  strcpy((players+i)->player\_name,temppname);  printf("Enter team name : ");  scanf(" %[^\n]%c",temptname);  (players+i)->team\_name=(char \*)malloc(50);  strcpy((players+i)->team\_name,temptname);  printf("Enter batting average :");  scanf("%d",&((players+i)->batting\_average));  }  }  void teamsort()  {  int p,c;  p=0;  while(p<5)  {  *// printf("halo");*  c=p; *//using insertion sort where p is number of passes and c is number of comparisions*  for(int i=c;i>0;i--)  {  if(strcmp(players[i-1].team\_name,players[i].team\_name)>0)  {  printf("halo");  struct cricket temp;  temp=\*(players+i); *//member-wise copy*  \*(players+i)=\*(players+(i-1));  \*(players+(i-1))=temp;  }  }  p++;  }  }  void print5(){  printf("Player list after team wise sorting :\n");  for(int i=0;i<5;i++)  {  printf("%s\t%s\t%d\n",(players+i)->player\_name,(players+i)->team\_name,(players+i)->batting\_average);  }  }  int main(){  read5();  teamsort();  print5();  } |
|  | **Output Screenshot:**  **1** |
| **2** | **Implement Priority Queue using an Unordered Linked list.**  Write functions for the following  1)Initialization  2)Enqueue  3)Dequeue  4)Display  **Output:**  enter ua choice  1.insert 2.delete 3.display 4 exit  1  enter the detail and priority  10  1  enter ua choice  1.insert 2.delete 3.display 4 exit  1  enter the detail and priority  20  2  enter ua choice  1.insert 2.delete 3.display 4 exit  1  enter the detail and priority  30  3  enter ua choice  1.insert 2.delete 3.display 4 exit  3  30 3  20 2  10 1  enter ua choice  1.insert 2.delete 3.display 4 exit  1  enter the detail and priority  40  0  enter ua choice  1.insert 2.delete 3.display 4 exit  3  40 0  30 3  20 2  10 1  enter ua choice  1.insert 2.delete 3.display 4 exit  2  deleted node detail is 30 with priority 3  enter ua choice  1.insert 2.delete 3.display 4 exit  2  deleted node detail is 20 with priority 2  enter ua choice  1.insert 2.delete 3.display 4 exit  2  deleted node detail is 10 with priority 1  enter ua choice  1.insert 2.delete 3.display 4 exit  2  deleted node detail is 40 with priority 0  enter ua choice  1.insert 2.delete 3.display 4 exit  2  no elements to delete  enter ua choice  1.insert 2.delete 3.display 4 exit  4 |
|  | **Program:**  **2prog.c:**  #include "2\_1.h"  int main(){  *//intial start initialisation*  struct node\* head=(struct node\*)malloc(sizeof(struct node));  head->next=NULL;  head->value=-1;  int flag=1;  int choice;  while(flag)  {  printf("--------------------\n");  printf("1.Insert \n");  printf("2.Delete \n");  printf("3.Display \n");  printf("4.Exit \n");  printf("Enter your option :");  scanf("%d",&choice);  switch(choice)  {  case 1:  {  int value,priority;  printf("Enter value :");  scanf("%d",&value);  printf("Enter priority :");  scanf("%d",&priority);  nodeinsert(head,value,priority);  break;  }  case 2:  {  nodedelete(head);  break;  }  case 3:  {  nodedisplay(head);  break;  }  case 4:  flag =0;break;  default:  printf("Invalid option \n");  }  }  }  **2\_1.h:**  #include<stdio.h>  #include<stdlib.h>  struct node{  int value;  int priority;  struct node\* next;  };  void nodeinsert(struct node\* *head* , int *value* , int *priority*);  void nodedisplay(struct node\* *head*);  void nodedelete(struct node\* *head*);  **2\_1.c:**  #include "2\_1.h"  void nodeinsert(struct node\* *head* , int *value* , int *priority*)  {  if(head->next==NULL) *//if first element*  {  struct node\* temp=(struct node\*)malloc(sizeof(struct node));  temp->value=value;  temp->priority=priority;  head->next=temp;  temp->next=NULL;  return;  }  else{  struct node\* previous=head;  struct node\* current=head->next;  while(current->next!=NULL) *//adding elements in the middle*  {  if(current->priority>priority)  {  struct node\* temp=(struct node\*)malloc(sizeof(struct node));  temp->value=value;  temp->priority=priority;  previous->next=temp;  temp->next=current;  return;  }  else{  previous=current;  current=current->next;  }  }  if(current->priority>priority) *//if middle did not work adding in last or second last*  {  struct node\* temp=(struct node\*)malloc(sizeof(struct node));  temp->value=value;  temp->priority=priority;  previous->next=temp;  temp->next=current;  return;  }  else{  struct node\* temp=(struct node\*)malloc(sizeof(struct node));  temp->value=value;  temp->priority=priority;  current->next=temp;  temp->next=NULL;  return;  }  }  }  void nodedelete(struct node\* *head*)  {  if(head->next==NULL)  printf("The node is empty!\n");  else{  struct node\* deletednode=head->next;  struct node\* newtop=deletednode->next;  head->next=newtop;  printf("Deleted node with value:%d and priority:%d \n",*deletednode*->value,*deletednode*->priority);  }  }  void nodedisplay(struct node\* *head*)  {  if(head->next==NULL)  {  if(head->value==-1)  printf("Start of list->");  else  printf("%d %d->",*head*->value,*head*->priority);  printf("End of list! \n");  return;  }  if(head->value==-1)  printf("Start of list ->");  else  printf("%d %d->",*head*->value,*head*->priority);  nodedisplay(*head*->next);  } |
|  | **Output Screenshot:**  **2_1**  **2_2**  **2_3**  **2_4** |