COMP 2401
Test #2

1. [2 marks] **Answer**: b
2. [2 marks] **Answer**: a
3. [2 marks] **Answer**: d
4. [2 marks] **Answer**: c
5. [2 marks] **Answer**: d
6. [2 marks] **Answer**: c

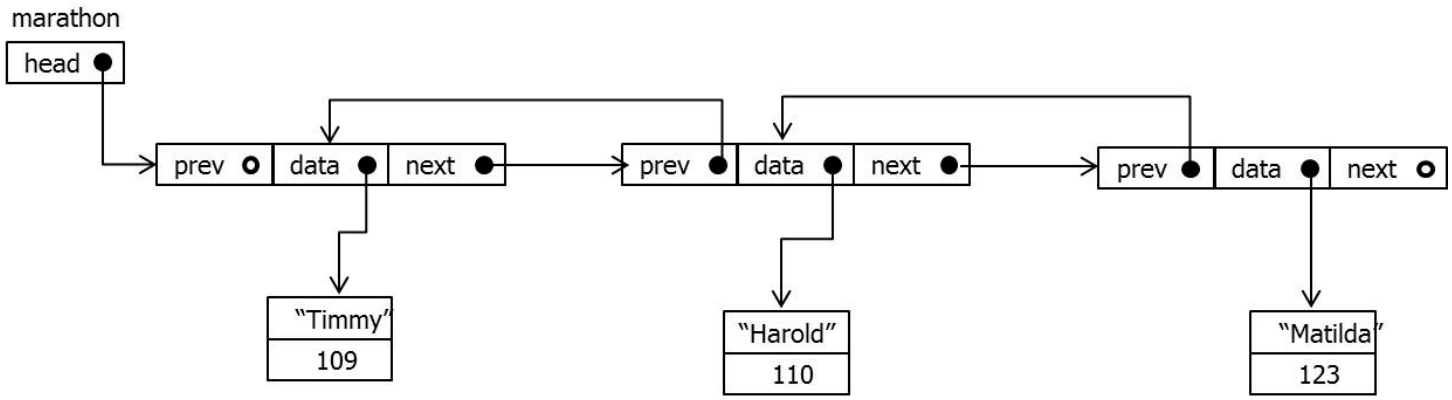
7. [8 marks]

```
void  foo ( int **p )
{
   *p  =  ( int  * )  malloc ( sizeof ( int ) ) ;
   **p =  20 ;
}

int main()
{
  int * x = NULL;
  foo ( &x ) ;
  printf("%d\n", *x); free(x);
  return 0;
}
```

Answer:
-- see code above
Marking:
-- 2 marks for making foo parameter a double pointer
-- 1 mark for dereferencing p in foo's first statement
-- 1 mark for dereferencing p twice in foo's second statement
-- 2 marks for passing address of x to foo from main
-- 2 marks for freeing x after printf

8. [30 marks]
   a. [6 marks]
      Answer:



Marking:

-- 1 mark for correct pointer to the head, -1 for pointer to tail
-- 1 mark for first node's previous set to NULL
-- 1 mark for last node's next set to NULL
-- 1 mark for correct next pointers (both of them)
-- 1 mark for correct prev pointers (both of them)
-- 1 mark for correct pointers to data, in correct order

b. [24 marks]

```
void addRunner(RunnerType *runner, ListType *list)
{
  NodeType *newNode, *currNode, *prevNode;
// 6 marks for allocating and initializing node
// -- 2 marks for malloc
// -- 1 mark for allocating correct size
// -- 1 mark for initializing data
// -- 2 marks for initializing prev and next
  newNode = (NodeType *) malloc(sizeof(NodeType));
  newNode->data = runner;
  newNode->prev = NULL;
  newNode->next = NULL;
  currNode = list->head;
  prevNode = NULL;    // this is necessary for adding to last position

// 4 marks for finding correct insertion point
// -- 2 marks for correct loop
// -- 2 marks for comparison and correct break point
  while (currNode != NULL) {
    if (newNode->data->finishTime < currNode->data->finishTime)
      break;
    prevNode = currNode;
    currNode = currNode->next;
  }

// 4 marks for setting head if adding to first position
  if (prevNode == NULL) {   // add to first position
    list->head = newNode;
  }
// 4 marks for setting newNode's prev and prevNode's next
  else {      // add to middle
    newNode->prev  = prevNode;
    prevNode->next = newNode;
  }

// 2 marks for setting newNode's next
  newNode->next  = currNode;

// 4 marks for setting currNode's prev
// -- 2 marks for checking for null currNode
// -- 2 marks for setting currNode's prev
  if (currNode != NULL)
    currNode->prev = newNode;
}
```

